INFOCLUB

REVISTA

DE INFORMATICA SI CALCULATOARE

2 1991



IDG INTERNATIONAL DATA GROUP

CUPRINS
CONTENTS

©INFOCLUB 2/91

5 **Flash:** Servicii moderne de comunicaţie; Turbo-Pascal, versiunea 6.0

9 **Şah:** Şah-computer (II)

14 **Laborator Spectrum:** Program pentru analiza spaţiilor într-un text

18 **Actualitatea PC:** Cursoare grafice

23 **Spot:** Tastatura PC

27 **Ghidul utilizatorului:** Turbo-Pascal, versiunile 5.0 şi 5.5 (II); MS-DOS (II)

32 **Computerworld:** Borland C++ 2.0; Dincolo de limitele unei reţele locale; Diferenţa între memoria expandată şi cea extinsă

34 **Ghidul utilizatorului:** PC 386 SX; Imprimante laser

39 **Top info:** SQL — nimic mai simplu!

40 **Macworld:** Mac vs PC

Revistă trimestrială de informatică şi calculatoare

ANUL II — NUMĂRUL 3

ADRESA: Piaţa Presa liberă nr.1, 79781 Bucureşti

TELEFON: 17 72 44 sau 17 60 10, interior: 1151-1258

Comitetul Director

Ioan ALBESCU
Gheorghe BADEA
Mihaela GORODCOV

Colegiul Ştiinţific

Dr. mat. Stelian NICULESCU (Ministerul Învăţămîntului şi Ştiinţei); dr. ing. Nicolae ŢĂPUŞ şi dr. ing. Valeriu IORGA (Institutul Politehnic Bucureşti, Fac. de Automatică); cercet. ing. Eugen GEORGESCU şi cercet. Ion DIAMANDI (Institutul de Tehnică de Calcul).

Corectură:
Lia Decei
Daly Dinu

Prezentare grafică:
Gabi Cătălinoiu
Maria Munteanu

Administraţia:
Editura „Presa Naţională"

Tiparul:
Întreprinderea „Arta Grafică"

Abonamentele se pot efectua pe adresa redacţiei prin mandat poştal pe numele Badea Gheorghe, urmînd ca abonamentul să fie expediat prin poştă la adresa indicată.

Pentru cititorii din instituţii, unităţi de învăţămînt, întreprinderi de stat şi particulare numărul minim de abonamente este de 50 exemplare/apariţie, putînd beneficia, în acest fel, de o reducere de 20%. Expedierea „abonamentului colectiv" se va face de către redacţie prin colet poştal la adresa indicată.

Preţ de vînzare: 35 lei

INFOCLUB este o publicaţie a International Data Group (IDG), cel mai mare editor de reviste de informatică şi calculatoare din lume. În fiecare lună, 25 de milioane de oameni citesc una sau mai multe publicaţii IDG.

Publicaţiile IDG includ: ARGENTINA: Computerworld Argentina; ASIA: Computerworld Hong Kong, Computerworld Southeast Asia, Computerworld Malaysia, Computerworld Singapore, Infoworld Hong Kong, Infoworld SE Asia; AUSTRALIA: Computerworld Australia, PC World, Macworld, Lotus, Publish; AUSTRIA: Computerwelt Oesterreich; BRAZILIA DataNews, PC Mundo, Automacao and Industria; BULGARIA: Computerworld Bulgaria, Computer Magazine; CANADA: ComputerData, Direct

RĂSFOIND PRESA INTERNAȚIONALĂ...

...ai un sentiment straniu. Îți propui de la început să urmărești o anumită temă, un anumit produs sau idee. Și constați că, plecînd de la bazele de date, ajungi la rețele, te lași furat de fascinația și multitudinea profesiilor informatice sau, inevitabil, ajungi la ghidul cumpărătorului.

Ce să alegi? Zeci de titluri și publicații din multe țări ale lumii te ispitesc: INFOWORLD, INFO PC, PC, WORLD, LE MONDE INFORMATIQUE, PUBLISH, COMPUTERWORLD (inclusiv cel pentru Bulgaria), MACWORLD, NETWORK WORLD, PC GAMES, GAME PRO, TELECOMS, AMIGA WORLD, BUYER'S GUIDE, DIGITAL NEWS și multe, multe altele care înseamnă informație, educație și integrare într-un circuit internațional. Cuprinderea acestei cantități uriașe de informație într-o revistă destul de subțire și cu apariție trimestrială (ambele din motive obiective pe care cred că le bănuiți și care vor face poate obiectul unui alt editorial) este practic imposibilă. Iar selecția obiectivă a informației, la fel.

Pentru acest număr am selecționat cîteva informații legate de piață, grupate sub genericul „Ghidul cumpărătorului", care vin, sperăm, în ajutorul utilizatorilor, în sensul de a se orienta asupra caracteristicilor tehnice și asupra prețurilor de producător la unele tipuri de calculatoare și imprimante laser care pătrund masiv în clipa de față în țara noastră.

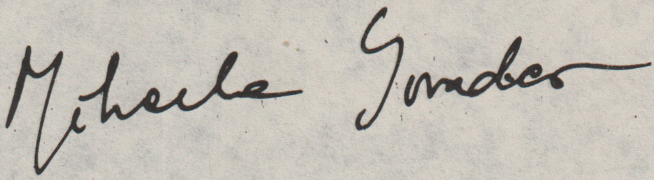
De asemenea, vă propunem o comparație între două importante tipuri de calculatoare: MAC și PC, care, de fapt, au făcut și obiectul copertei noastre, dată fiind importanța subiectului. Așadar, „MAC vs PC" cu caracteristici și tabele comparative extrase dintr-un număr recent al revistei MACWORLD. Dincolo de aceste orientări majore, am abordat, după cum veți vedea, și cîteva noutăți care au scopul de a preveni și pregăti publicul în momentul în care aceste programe și echipamente vor pătrunde masiv în țara noastră.

Așadar, în afară de toate aceste noutăți, noi am păstrat, firește, și rubricile tradiționale, de succes, ale revistei INFOCLUB, a căror pondere am încercat să nu o diminuăm, avînd în vedere spectrul larg de probleme la care trebuie să facem față.

Și acum puțin despre viitorul apropiat. Pentru aceasta avem nevoie de ajutorul dumneavoastră. Ce ați dori să citiți în INFOCLUB? Documentația pe care o primim de la IDG, precum și faptul că deținem exclusivitatea folosirii ei, ne îndreptățește să afirmăm că putem aborda orice subiect, de la PC la mașini RISC, de la educație pînă la sfaturi pentru alegerea unui fax, de la rețele pînă la cele mai diverse echipamente periferice. Ne-ați ajuta mult în selecția materialelor dacă ne-ați scrie preferințele dumneavoastră, cu atît mai mult cu cît pentru această toamnă vă pregătim și unele surprize, sperăm foarte plăcute.

Și încă ceva. După cum ați văzut, avem o rubrică intitulată „Dialog cu cititorii". Am primit deja multe scrisori din București și din țară (semn bun că revista ajunge) și le mulțumim tuturor pentru aprecieri și critici. Am dori mult ca această rubrică să fie nu numai una de dialog al redacției cu cititorii, ci, mai ales, o legătură, o conexiune între cititori, un fel de „Club INFO", deoarece, sînt convinsă, formăm o familie și avem multe să ne spunem.

Invităm încă o dată specialiștii și firmele de profil să ne țină la curent cu realizările și cu reușitele lor, pentru că, așa cum am mai spus-o, orice informație de la dumneavoastră este pentru întreaga lume!



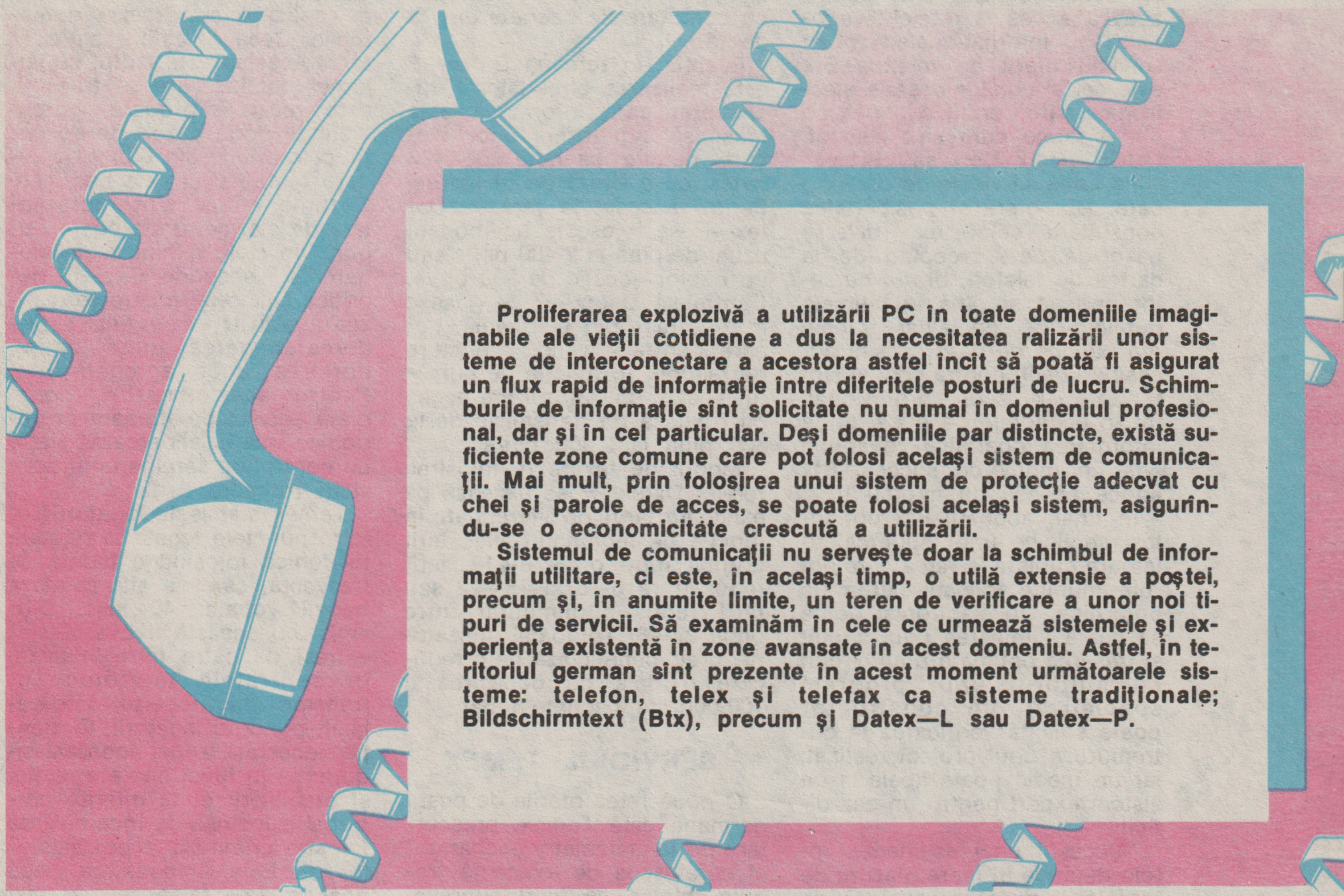
Access, Graduate CW, Macworld: CHILE: Informatica, Computacion Personal: COLUMBIA: Computerworld, Columbia: CEHOSLOVACIA: Computerworld Cehoslovacia, PC World: DANEMARCA: CAD/CAM WORLD, Computerworld Danmark, Communication World, PC World, Macworld, Unix World, PC LAN World: FINLANDA: Mikro PC, Tietoviikko, Tietotekniika: FRANȚA: Le Monde Informatique, Distributique, InfoPC, Telecoms International: GERMANIA: Computerwoche, Information Management, Amigawelt, PC Woche, PC Welt, Unix Welt, Macwelt RD: GRECIA: Computerworld, PC World, Macworld, Infoworld: UNGARIA: Computerworld SZT, Mikrovilag: INDIA: Computers and Communications: ISRAEL: People and Computers: ITALIA: Computerworld Italia, PC World Italia: JAPONIA: Computerworld Japan, Macworld: COREEA: Computerworld, PC World: MEXIC: Computerworld Mexico, PC Journal: OLANDA: Computerworld Netherland, PC World, Amiga World: NOUA ZEELANDĂ: Computerworld New Zealand, PC World New Zealand: NIGERIA: PC World Africa: NORVEGIA: Computerworld Norge, PC World Norge CAD/CAM, Macworld Norge: CHINA: China Computerworld, China Computerworld Monthly: FILIPINE: Computerworld Phillippines, PC Digest/PC World: POLONIA:Komputers Magazine, Computerworld: **ROMÂNIA: Infoclub;** SPANIA: CIM World, Communicaciones World, Computerworld Espana, PC World, Amiga World: SUEDIA: ComputerSweden, PC/Nyheterna, Mikrodatorn, PC World, Macworld: ELVEȚIA: Computerworld Schwitzerland, Macworld: TAIWAN: Computerworld Taiwan, PC World, Publish: THAILANDA: Computerworld: TURCIA: Computerworld Monitor, PC World/Turkiye: MAREA BRITANIE: Graduate Computerworld, PC Business World, ICL Today, Lotus UK, Macworld U.K.: STATELE UNITE: Amiga World, A+, CIO, Computerworld, Digital News, Federal Computer Week, GamePro, IDG Books, InfoWorld, Macworld, NextWorld, Network World, PC Games, PC World, Portable Office, PC Letter, Publish, Run, Sun Tech Journal: URSS: MIR PC, Computerworld URSS, Network, Manager Magazine: VENEZUELA: Computerworld Venezuela, Micro Computerworld: IUGOSLAVIA: Moj Mikro.

FLASH



Cristian IONESCU

SERVICII MODERNE DE COMUNICAŢIE PENTRU UTILIZATORII DE PC



Proliferarea explozivă a utilizării PC în toate domeniile imaginabile ale vieţii cotidiene a dus la necesitatea ralizării unor sisteme de interconectare a acestora astfel încît să poată fi asigurat un flux rapid de informaţie între diferitele posturi de lucru. Schimburile de informaţie sînt solicitate nu numai în domeniul profesional, dar şi în cel particular. Deşi domeniile par distincte, există suficiente zone comune care pot folosi acelaşi sistem de comunicaţii. Mai mult, prin folosirea unui sistem de protecţie adecvat cu chei şi parole de acces, se poate folosi acelaşi sistem, asigurîndu-se o economicitate crescută a utilizării.

Sistemul de comunicaţii nu serveşte doar la schimbul de informaţii utilitare, ci este, în acelaşi timp, o utilă extensie a poştei, precum şi, în anumite limite, un teren de verificare a unor noi tipuri de servicii. Să examinăm în cele ce urmează sistemele şi experienţa existentă în zone avansate în acest domeniu. Astfel, în teritoriul german sînt prezente în acest moment următoarele sisteme: telefon, telex şi telefax ca sisteme tradiţionale; Bildschirmtext (Btx), precum şi Datex—L sau Datex—P.

SISTEMELE TRADIŢIONALE DE COMUNICAŢII

În ceea ce priveşte telefonul, telexul şi telefaxul, modul de utilizare este foarte asemănător. Prin intermediul unui cuplor specializat şi al unui pachet software se realizează legături punct la punct fie cu alţi abonaţi, fie cu cutii poştale electronice. În acest sens, aici întîlnim experienţa deja acumulată în Statele Unite în cadrul reţelei BIX promovată de revista Byte. Prin folosirea unui PC în locul tradiţionalelor aparate specifice, se asigură o creştere a inteligenţei pusă la capătul firului cu consecinţa apariţiei unor noi servicii la postul abonat în sine.

Interesante sînt însă serviciile pe care le oferă sistemul de comunicaţii în sine, cu alte cuvinte organul de poştă respectiv.

SISTEMUL BILDSCHIRMTEXT

Sistemul Bildschirmtext (Btx) numără deja peste 200 000 abonaţi care îl folosesc în scopuri majoritar profesionale — peste 70% — pentru a comunica, a căuta în bănci de date, a transmite telexuri sau pentru a face comenzi la producătorii care îşi fac reclamă prin Btx. Motivul abundenţei utilizatorilor este constituit de costurile reduse implicate de legarea unui PC la reţea: un modem uzual de 1 200/75 biţi/s şi un decodor software sau un adaptor special pus la dispoziţie de poştă. Dat fiind că utilizatorii schimbă între ei doar texte şi o pseudografică, costul terminalelor, precum şi costul transmisiei propriu-zise sînt suficient de reduse. Profesional, acest sistem convine atît micilor întreprinderi atrase de nivelul redus al investiţiilor necesare, cît şi marilor companii care pot comod şi ieftin să-şi extindă sistemul de comunicaţii.

Pachetul software existent în staţia Btx poate asigura o gamă foarte largă de servicii, funcţie

de gradul său de complexitate și de cerințele beneficiarului. Serviciile oferite de poștă, mai ales legătura la rețeaua telex și de telefax, devin deosebit de valoroase pentru abonații la Btx pentru că astfel ei nu mai trebuie să achiziționeze suplimentar aceste echipamente pentru a avea acces la rețelele naționale sau internaționale. Un alt serviciu oferit de rețeaua Btx este cel al cutiilor poștale electronice care permit cuplarea cu alte sisteme cum sînt Geonet, UUCP (Unix Net) sau Bitnet.

Pe lîngă serviciile de comunicare, Btx oferă și posibilitatea consultării celor mai diverse baze de date începînd de la cartea de telefon, orarul curselor aeriene și pînă la ultimele publicații de specialitate. Unele din băncile de date nu sînt accesibile decît contra cost și sînt utilizate în scop profesional. Într-o economie de piață, unde informarea operativă constituie des un motiv de succes, Btx aduce servicii de neprețuit.

În final, abonații Btx pot folosi serviciile unor supercalculatoare cuplate la rețea care pot rula anumite aplicații extrem de puternice. Un electronist poate să ceară simularea unui circuit și trasarea optimă a unui circuit cu ajutorul unui program specializat, un constructor poate solicita verificarea la cutremure a unui proiect realizat, iar un medic poate apela la un sistem expert pentru un caz dificil.

Succesul unei asemenea rețele depinde în mare măsură de tarifele impuse, precum și de diferențierea tarifelor în funcție de momentul zilei în care este folosit Btx. După cum se vede, prin adoptarea unor tarife diferențiate se poate ajunge la o încărcare uniformă a rețelei telefonice.

SERVICIUL DATEX—P

Un al doilea serviciu, Datex—P, este oferit în mod special pentru comunicații între calculatoare. Datex—P este mai avantajos decît Btx prin viteza mare de lucru, reducerea erorilor de transmisie și o structură mai convenabilă a costurilor.

Peste 45 000 de conexiuni în RFG la sfîrșitul anului 1989 dovedesc utilitatea acestui sistem de transmisiuni cu comutare de pachete. La el este cuplată și rețeaua de comunicații pe linii închiriate Datex—L.

Avantajele oferite de Datex—P sînt: costuri independente de distanța parcursă de pachetele de informație; adaptarea diferitelor viteze de comunicație între parteneri; accesul la rețeaua telefonică prin modem sau cuplor acustic; acces la 160 de rețele de comunicație cu comutare de pachete din 75 de țări.

Esențial în rețeaua Datex—P este realizarea circuitelor virtuale prin care informația este tansmisă sub forma unor blocuri, numite pachete, de 128 bytes, cu o viteză de 64 kbiți/s. Pentru accesul la rețeaua Datex—P se folosește un modem uzual de pînă la 2 400 biți/s sau un cuplor acustic de 300 biți/s. Centralele Datex—P se găsesc în 18 orașe din Germania, fiecare centrală avînd 4 numere de telefon. În funcție de numărul apelat, utilizatorul are acces cu o viteză de transmisie de la 300 pînă la 2 400 biți/s.

Modul de taxare în folosirea rețelei Datex—P se bazează pe existența tarifelor diferențiate în funcție de momentul zilei: tarif normal între orele 8—18, tarif redus între 6—8 și 18—22 sau tarif redus suplimentar între orele 22—6. Totodată, utilizatorii „grei" beneficiază de o reducere o dată cu depășirea a 100 000 de pachete lunar.

SERVICIUL TEMEX

O nouă rețea oferită de poșta germană este Temex, prescurtare de la „telemetry exchange" al cărei sens de existență trebuie înțeles ca fiind supraveghere și control la distanță. Temex se bazează pe rețeaua telefonică deja instalată ca mijloc de vehiculare a informației și pe PC ca stații terminale.

Să dăm un exemplu de utilizare! Într-o casă particulară este instalată baza tehnică a unui sistem de supraveghere și control. Cînd părăsește casa, stăpînul pune în funcțiune alarma și programează ora de începere a reîncălzirii camerelor, ținînd cont de ora de reîntoarcere acasă. În timpul zilei, el poate cere, prin intermediul computerului său, să i se comunice care este stare locuinței: temperaturi, dacă au fost apeluri telefonice sau de cîte ori s-a deschis ușa de la intrare. Dacă după-amiază el se întoarce mai tîrziu de la serviciu, poate decala corespunzător ora de începere a încălzirii casei. Alternativ, dacă în casă are loc un eveniment neobișnuit: alarmă la inundație, la fum sau la spargere, centrala de supraveghere poate alarma poliția și pe stăpînul casei folosind rețeaua Temex.

Fără îndoială că exemplele de aplicații sînt foarte numeroase. Ceea ce este esențial în toate acestea este faptul că volumul datelor care se transmit este redus. Totodată, putem distinge două cazuri de aplicații: cele critice din punct de vedere temporal și cele necritice. Cazurile critice sînt reprezentate de alarme, unde poșta asigură un ciclu minim de interogare de 7 secunde. Cazurile necritice sînt cele în care se realizează activități de rutină: aprinderea/stingerea unor lumini, pornirea unor agregate etc. Pentru aceste cazuri, poșta oferă servicii de abonament periodice, care eliberează stația de control de sarcina unor activități rutiniere.

Temex stabilește legături fixe între punctele legate la rețeaua telefonică folosind o bandă de frecvență, care se află în afara benzii vocale: 40 kHz. Biții emiși cu această frecvență sînt extrași de către filtre instalate în centralele telefonice și transmiși mai departe către celălalt punct al legăturii. O atenție deosebită a fost acordată siguranței în funcționare a rețelei și stabilității ei la diferite perturbații, inclusiv la încercări de folosire criminală a posibilităților ei. Este limpede că dacă apar mesaje false sau, și mai grav, nu se anunță alarme, consecințele social-economice sînt enorme.

Și în cazul rețelei Temex, politica prețurilor stabilită de poșta germană este decisivă pentru succesul serviciului oferit. Pe lîngă prețuri, existența unor centrale de supraveghere a locuințelor reprezintă factorul principal de răspîndire a acestei rețele. În acest domeniu a început deja o concurență aprigă între diferitele firme de producere a aparatajului electronic.

Iată deci aplicații uzuale care au menirea să ne convingă încă o dată că informatica și calculatoarele în general trăiesc și progresează exclusiv prin intermediul soluțiilor care le oferă. Și aici, trebuie să recunoaștem, nu există practic limită.

Adriana POPESCU

TURBO-PASCAL VERSIUNEA 6.0

Intrat deja în uzul programatorilor din lumea întreagă, Turbo-Pascalul continuă să se mențină în topul limbajelor de programare, datorită flexibilității, modularității, modului facil de folosire, adecvat unei game largi de aplicații, în cele mai diverse domenii. Destinat posesorilor de calculatoare IBM PC, IBM PS/2 și compatibile, Turbo-Pascal este un limbaj structurat, de nivel înalt, oferind utilizatorilor posibilitatea dezvoltării de aplicații de orice natură și cu un grad mare de complexitate. Integrîndu-se în contextul general al proiectării de medii de programare din ce în ce mai performante și mai comode pentru utilizator, firma Borland a lansat de curînd pe piață Turbo-Pascal 6.0, limbaj orientat pe obiecte, avînd o serie de particularități care îl distanțează de versiunile anterioare.

La o privire superficială, diferențele par minore, însă aparențele sînt înșelătoare, ascunzînd prefaceri interne. Se cuvine astfel să amintim că noua versiune, grație bibliotecii profesionale compatibile TP 6, facilitează refolosirea modulelor scrise și testate, oferind peste 100 de tipuri de obiecte. În al doilea rînd, mediul de dezvoltare a aplicațiilor are o interfață amintind de cea a Turbo-C ++ cu meniuri, ferestre de dialog, mouse, editor multifișier care poate edita fișiere pînă la 1 Mb, facilități de depanare, de testare pas cu pas a programelor, salvare și restaurare a spațiului de lucru. Circumscris acestui mediu de dezvoltare (păstrînd deci toate caracteristicile acestuia), firma oferă un mediu de dezvoltare a aplicațiilor cu metode de programare bazate pe evenimente, care conține ansamblul de biblioteci necesare în exploatarea programelor, avînd unele trăsături specifice, cum ar fi: modificări de sintaxă ce permit tratarea funcțiilor ca proceduri (ignorînd rezultatele primelor), noi cîmpuri și metode de declarare a obiectelor, un manager de stivă mai rapid și care reduce fragmentarea, help-uri foarte bine documentate pentru orice funcție sau procedură a bibliotecii.

Documentația livrată de firmă conține patru manuale. Ghidul utilizatorului, oferă informații despre instalarea și folosirea mediului de programare Turbo-Pascal, programarea orientată pe obiecte, depanarea, compilarea programelor, precum și lucrul cu editorul. Ghidul programatorului cuprinde aspecte tehnice ale limbajului, conținutul bibliotecilor, legăturile cu asamblorul, mesaje de eroare, explicații ale directivelor compilatorului. Manualul bibliotecii conține referințe ale tuturor funcțiilor și procedurilor din biblioteca TP 6. În sfîrșit, manualul Turbo-Vision prezintă într-o formă unitară mediul de programare sus-amintit.

Pentru că multe din trăsăturile limbajului TP 6 se întîlnesc în variantele anterioare, TP 5.0 și TP 5.5, vom încerca să punem în evidență doar aspectele noi ale acestei versiuni.

Editorul reprezintă o schimbare vizibilă față de versiunile anterioare permițînd editarea de texte (căutări, inserări, modificări, poziționări, ștergeri etc., precum și operații cu blocuri) în mod ecran, folosind ferestre, meniuri accesibile atît de la tastatură, cît și mouse, avînd totodată help-uri bine documentate corespunzătoare fiecărei operații în parte. Puterea lui rezidă în cele aproximativ 50 de comenzi ce se pot grupa în patru categorii (legate de cursor, operații de inserare și ștergere, operații cu blocuri, diverse). Acest nou editor permite o structură de parametri servind la extensia fișierelor, autoindentificarea lor și aplicații în mod ecran, gestiunea a nouă mărci corespunzătoare memorării unor poziții pe ecran, ce pot fi regăsite ulterior (Ctrl/K 1-9 Ctrl/Q 1-9) prin intermediul funcțiilor SetMark, GotoMark, trăsătură esențială în lucrul cu ferestre. Este posibilă de asemenea inserarea unei mărci (caracter de control), cu ajutorul comenzii Ctrl/P (comandă inspirată din WordStar), precum și folosirea tastelor F7 și F8 corespunzînd execuției pas cu pas a programului, respectiv execuției pasului următor din procedura curentă, sau poziționarea în susul ori josul ecranului (Ctrl Home și Ctrl Finn) fără defilarea textului.

Ultima instrucțiune realizează revenirea la starea inițială. Această buclă reprezintă baza aplicațiilor, orice eveniment fiind interpretat.

Înainte de a prezenta modul în care este tratat un eveniment simplu, ca deplasarea cursorului, să amintim în treacăt cele mai importante obiecte din T-V. Acestea sînt organizate într-o ierarhie care are în vîrf TObject, TPoint, TRect. Natural, în T-V și ferestrele sînt obiecte derivate din TWindows, care alături de TView, TGroup, TDesktop, TProgram, TApplication, TDialog și multe altele reprezintă cîteva dintre tipurile primitive de obiecte, structura acestora fiind arborescentă.

Deplasarea cursorului este un eveniment primit de obiectul TEditor. HandleEvent, prin intermediul TView. HandleEvent. Rutina ConvertEvent interpretează comanda, extrăgînd din anumite tabele (FirstKeys, QuickKeys și BlockKeys) numărul comenzii pe care trebuie să o genereze (numărul corespunzător tastei). Adresele din aceste tabele sînt memorate în tabela KeyMap. Un subprogram, ScanKeyMap, examinează una din aceste tabele (a cărei adresă e furnizată ca parametru), pentru a găsi caracterul și a-l analiza, întorcînd codul comenzii respective sau 0, dacă acesta nu e găsit. De menționat că în timpul interpretării ScanKeyMap tratează și corectează informațiile. Remarcăm astfel că Turbo-Pascal versiunea 6.0 răspunde sub toate aspectele cerințelor utilizatorilor familiarizați sau nu cu programarea evenimențială, pregătindu-i totodată cu un mod de gîndire foarte util pentru Windows.

Schimbarea de esență însă o constituie mediul de dezvoltare a aplicațiilor orientate pe obiect, Turbo-Vision. În interiorul acestui mediu programele se modifică prin extinderea lor, prin derivarea sau adăugarea unor tipuri noi de obiecte. De asemenea, mediul de programare trebuie privit ca o ierarhie de instrumente necesare programatorului. O aplicație în T-V începe cu instanțierea unui obiect descinzînd din TApplication. Acest obiect, TApplication, constituie învelișul programului, conținînd descrierea meniului, linia de stare și spațiul de lucru. Metoda de inițializare a aplicației apelează la metodele de inițializare ale obiectelor conținute.

De exemplu:

```
var
    HelloWorld: THelloApp
begin
        HelloWorld. Init;
        HelloWorld. Run;
        HelloWorld. Done;
end.
```

Prima instrucțiune este un apel de inițializare a variabilelor, de construire a obiectelor și a spațiului de lucru. A doua poate fi privită ca o buclă de gen repetă... pînă cînd..., a cărei semnificație poate fi descrisă în pseudocod astfel:

```
repeat
        Get an Event
        Handle the Event
until Quit.
```

Fiecare obiect integrat în aplicație posedă o metodă Handle Event a cărei încărcare constă în tratarea evenimentelor respective (deplasarea cursorului, de exemplu). La nivelul cel mai înalt aceste evenimente sînt interpretate pentru a furniza codurile comenzilor, simplificîndu-se astfel interpretarea diverselor rutine Handle Event.

ŞAH COMPUTER



Viorel DARIE

ŞAH-COMPUTER (II)

În numărul precedent al revistei, am început un mini-curs de programare a şahului pe calculator, invitînd totodată pe toţi cei care au posibilitatea şi pasiunea de a se ocupa de acest fascinant subiect să înceapă să-şi realizeze propriile programe de şah, cu care avem intenţia să organizăm campionate naţionale de şah, iar cele mai bune vor fi delegate să participe la concursuri internaţionale. De asemenea am făcut o simplă, însă lămuritoare formalizare a reprezentării tablei de şah, abordînd problema generării de liste şi de arbori de mutări, a algoritmului mini-max.

Considerăm că, pentru a putea înţelege expunerea care urmează, cititorul trebuie să revadă cele prezentate în primul articol al acestui serial din numărul precedent al revistei.

CRITERIILE DE REDUCERE DE RAMURI ÎN ARBORELE MUTĂRILOR

O dată cu creşterea numărului de semimutări pentru care se face generarea arborelui mutărilor asociate unei poziţii de pe tabla de şah, numărul de noduri (deci şi variante) în arborele generat creşte exponenţial, făcînd ca, chiar pentru un număr mic de semimutări, nici cele mai rapide calculatoare din lume să nu mai poată analiza toate variantele implicate.

Oare este necesar să fie analizate toate variantele pentru a alege mutarea optimă? Oare nu se poate rări mulţimea variantelor generate, cu scopul (evident) ca variantele generate să fie cît mai lungi, cu speranţa ca jocul să fie cît mai bun?

Abandonarea arbitrară a variantelor (şi orice programator de şah se poate convinge repede de acest lucru) conduce la un joc cu multe inexactităţi, căci, de multe ori, mutările abandonate sînt tocmai soluţiile optime, mutări obligatorii, apărări de la mat, rezolvarea precisă a unor situaţii dificile etc.

Metodele de reducere a variantelor, în procesul de determinare a variantei optime de joc, sînt de două tipuri: exacte (soluţia optimă nu este omisă, atît cît permite arborele pe o adîncime dată) şi **neexacte** (sau **euristice**, cînd soluţia optimă poate fi omisă, totuşi, de regulă, soluţia determinată este acceptabilă).

Prezentăm întîi cîteva metode exacte de reducere de variante.

Criteriul alfa-beta. Să presupunem că pe nivelul **k—1** la mutare este albul şi că pe acest nivel au fost analizate una sau mai multe mutări, cărora le corespunde o valoare mini-max parţială P_{k-1} (un maxim parţial). Prin urmare, pe nivelul **k**, sînt mutările negrului, care minimalizează.

Presupunem că, la un moment dat, se analizează o mutare x de la nivelul **k—1**, care a fost efectuată şi pentru care s-a generat o listă de mutări a negrului. Presupunem că pentru o mutare (analizată) y din această listă, valoarea mini-max este P_k. **Criteriul alfa. Dacă $P_{k-1} \geq P_k$, atunci se pot abandona, fără analiză, toate mutările negrului din lista de la nivelul k care au mai rămas neanalizate şi se poate trece la analiza următoarei mutări din listă de pe nivelul k—1.**

Într-adevăr, dacă $P_{k-1} \geq P_k$, atunci P_{k-1}, $\geq$ minimul valorilor mini-max pentru mutările din lista pe nivelul **k**, deci, în acest caz, pe nivelul **k—1**, unde se face maximul, valoarea P_{k-1} nu va fi îmbunătăţită. Deci analiza completă a listei de pe nivelul **k** ar fi o pierdere de timp, căci rezultatul va fi refuzat pe nivelul **k—1.** Cele spuse mai sus sînt ilustrate în **figura 6.**

Dacă pe nivelul **k—1** este la mutare negrul, prin analogie, cu aceleaşi notaţii, putem formula:

Criteriul beta. Dacă $P_{k-1} \leq P_k$, atunci se pot abandona, fără analiză, toate mutările albului din lista de la nivelul k, care încă n-au fost analizate, şi se poate trece la analiza următoarei mutări din lista de pe nivelul k—1.

Putem formula mai nuanţat principiul care rezultă din cele două criterii enunţate mai sus: **un jucător va abandona analiza unei variante cînd va constata că ea e mai slabă decît cea mai bună variantă pe care deja a analizat-o.**

Beneficiul adus de criteriul alfa-beta este imens pentru viteza cu care este analizat arborele de joc. Astfel, într-un joc cu lungimea ramurilor de 4 semimutări, numărul total de noduri se reduce de la circa 1 milion la numai circa 10 000 de noduri, deci de 100 de ori, chiar în condiţiile în care nu există o ordonare prealabilă a listelor de mutări. Criteriul alfa-beta funcţionează după prima, a doua etc. mutare analizată **pe un nivel.** Cu cît va funcţiona mai repede, cu atît beneficiul va fi mai mare, economisindu-se timp de calcul.

În momentul generării listelor de mutări, fiecărei mutări i se atribuie o **pondere statică**, calculată cît mai simplu, dar care să reflecte cît mai bine starea acelei poziţii. Se poate observa o puternică corelaţie între rangurile rezultate în urma sortării listelor de mutări conform **ponderilor statice** şi conform notelor **mini-max**, motiv pentru care analiza variantelor nu se face arbitrar, ci conform ordinii date de ponderile statice. Justificarea sortării constă tocmai în faptul că, ordonînd bine listele de mutări, criteriul alfa-beta va funcţiona cît mai la începutul listelor.

Criteriul alfa-beta trebuie aplicat şi în subprogramul de generare de mutări, pe măsură ce este generată cîte o mutare, căci, dacă apar **mutări terminale,** pentru ele valorile mini-max sînt chiar ponderile statice, iar cînd funcţionează criteriul alfa-beta, se abandonează generarea listei de mutări de pe acel nivel.

Criteriul alfa-beta se aplică consecvent, pe toate nivelurile. Evident, nu se poate aplica pe nivelul **k=1,** căci nu mai există nivelul **k—1** cu care să fie comparat. Prin urmare, pe nivelul 1, vor trebui să fie analizate toate mutările din lista respectivă. În figura 7 este ilustrat modul cum funcţionează criteriul alfa-beta.

Văzînd beneficiile deosebite pe care criteriul alfa-beta le aduce, sîntem îndemnaţi să exploatăm această resursă la maximum. În afara străduinţei de a ordona cît mai bine mutările conform ponderilor statice, oare ce se mai poate face pentru a stimula cît mai mult criteriul alfa-beta?

Criteriul alfa-beta-deep (de adîncime). S-ar putea ca, pe nivelul **k**, valoarea mini-max P_k a mutării curente, deja analizată, să fie acceptată de adversar pe nivelul **k—1**, dar să nu fie acceptată pe unele niveluri **k—3, k—5** etc. În acest caz, se poate renunţa la analiza mutărilor din lista pe nivelul **k** care încă n-au fost analizate.

Sugestiv, acest criteriu îl putem enunţa astfel: **nu are sens să analizezi variante mai slabe decît variantele parţial optime determinate pe nivelurile mai mici.**

Se observă că **alfa-beta-deep** funcţionează doar cînd arborele are variante de cel puţin 4 semimutări, altfel el se identifică cu **alfa-beta.** În figura 8, este ilustrat modul cum operează criteriul **alfa-beta-deep.** În loc de a compara de fiecare dată valoarea P_k cu toate valorile P_{k-1}, P_{k-3} etc. se procedează astfel: pe nivelurile $k > 2$ se iniţializează valorile mini-max, nu cu $-\infty$ (pentru alb) sau $+\infty$ (pentru negru), ci se iniţializează cu P_{k-2}.

Criteriile alfa-beta, ca şi **alfa-beta-deep,**

sînt criterii de reducere a variantelor din categoria celor **exacte**, prin care **mutarea optimă** și **varianta optimală de joc** nu sînt omise. Pînă una-alta, criteriul alfa-beta-deep (se vede că) include și criteriul alfa-beta, deci reduce cele mai multe variante, motiv pentru care este preferabil de implementat.

Particularități legate de alfa-beta-deep.

Deși, neîndoielnic, criteriul alfa-beta-deep reduce drastic numărul de variante din arborele mutărilor, totuși, arborele continuă să aibă caracter exponențial, dar factorul de multiplicare de la un nivel la altul este mult mai mic, deci numărul total de variante este sub forma unei baze mai mici ridicate la o putere, ceea ce permite ca lungimea variantelor să crească cu cîteva semimutări.

Există o perioadă de început a analizei arborelui cînd valorile cu care se inițializează valorile mini-max pe niveluri sînt $\pm\infty$, și cînd criteriul alfa-beta-deep nu funcționează, făcînd să fie necesară analiza tuturor mutărilor din listele pe nivelurile respective. Reducerile de variante (conform criteriului alfa-beta-deep) încep să funcționeze de-abia după ce valorile mini-max parțiale se actualizează. Adică, dacă $P_{k-1} = \mp\infty$, atunci, pe nivelul **k**, sîntem nevoiți să analizăm toate mutările, iar dacă $P_{k-1} \neq \mp\infty$, atunci reducerile încep cu a doua sau a treia etc. mutare din listă. Priviți figura 9.

Cele spuse mai sus nu pot fi evitate, oricîte străduințe s-ar face în ordonarea cît mai bună a listelor de mutări, conform ponderilor statice. Se estimează că, în cel mai bun caz, într-un arbore de lungime M a variantelor, în care există, în medie, N mutări în fiecare listă, numărul minim de variante efectiv generate este incomparabil mai mic decît astronomicul N^M. Dacă notăm cu Q **media listelor mutărilor efectiv analizate** pe toate nivelurile, atunci se generează un număr de variante mai mic decît Q^M. Q se numește **factor de ramificare.** Acest număr măsoară **eficiența** unui algoritm de analiză a variantelor din arborele mutărilor.

S-au elaborat programe foarte performante, pentru care factorul de ramificare este circa 3. În programele ATOM-64, ORIZONT-64, cu toate încercările, nu s-a obținut decît un factor de ramificare de circa Q = 7. E drept, obținerea unui factor de ramificare foarte mic presupune aplicarea unor algoritmi sofisticați, care consumă timp de calcul pentru evaluarea cît mai corectă a ponderilor statice. Efortul merită însă, căci avantajul reducerii de variante poate fi imens. Mai concret, dacă N = 30 de mutări, M = 10 semimutări, și dacă creștem de 10 ori timpul de calcul al ponderilor statice, și dacă reușim să trecem de la factorul de ramificare 7.0 la 3.0, atunci timpul de analiză al întregului arbore se reduce de:

$$(30 \cdot 7^9) : (10 \cdot 3^9 \cdot 30) \approx 200 \text{ ori.}$$

Criteriul alfa-beta-deep cu fereastră.

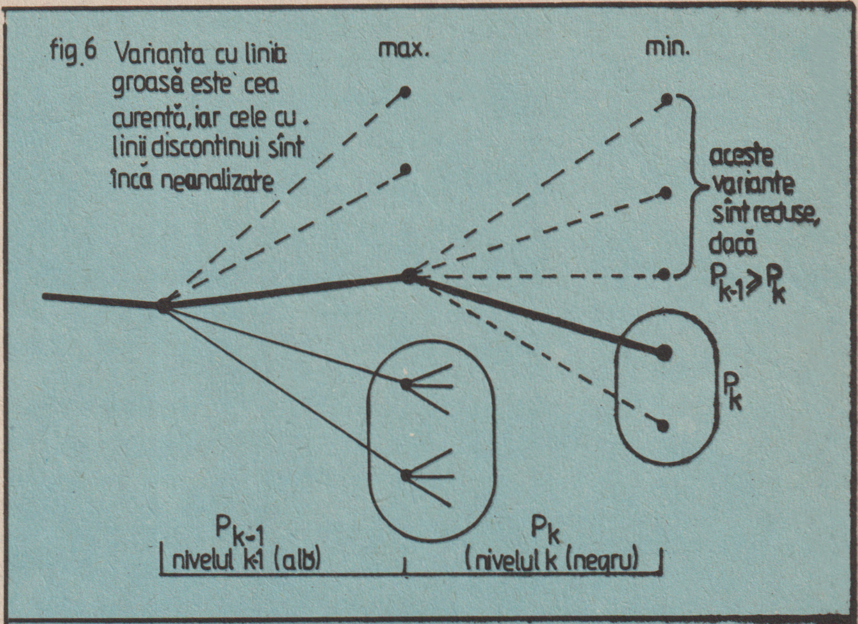
Sintetizînd cele spuse mai sus, dacă notăm $a_1, a_2, a_3, \ldots, a_n$ șirul valorilor mini-max parțiale pe nivelurile albului și cu: $b_1, b_2, b_3, \ldots, b_n$ șirul valorilor mini-max ale negrului, atunci constatăm următorul șir de inegalități:

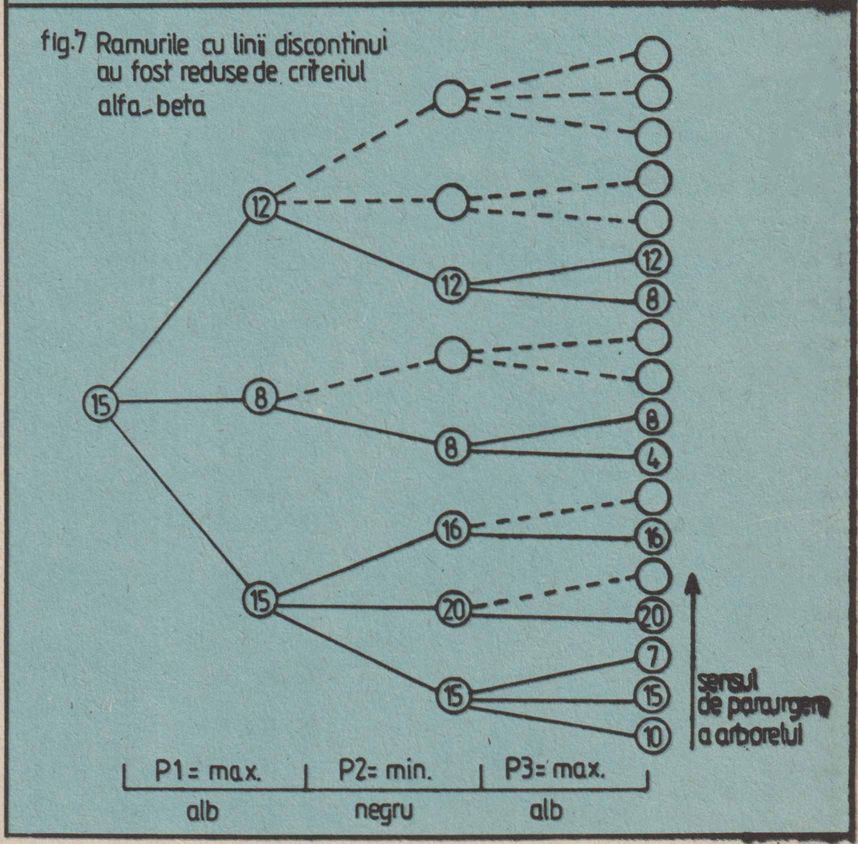
$$a_1 \leq a_2 \leq a_3 \leq \ldots_1 \leq a_n \leq b_n \leq b_{n-1} \leq \ldots \leq b_3 \leq b_2 \leq b_1,$$

ceea ce, mai sugestiv, se poate vedea din figura 10.

Conform notațiilor din paragrafele precedente, P_k este valoarea mini-max parțială pe nivelul **k**, deci:

$$P_k = \begin{cases} a_{k1} \text{ pe nivelul albului} \\ b_{k1} \text{ pe nivelul negrului} \end{cases}$$





unde $k1 = \frac{k}{2} + 1$, dacă k este impar și $k1 = \frac{k}{2}$, dacă k este par.

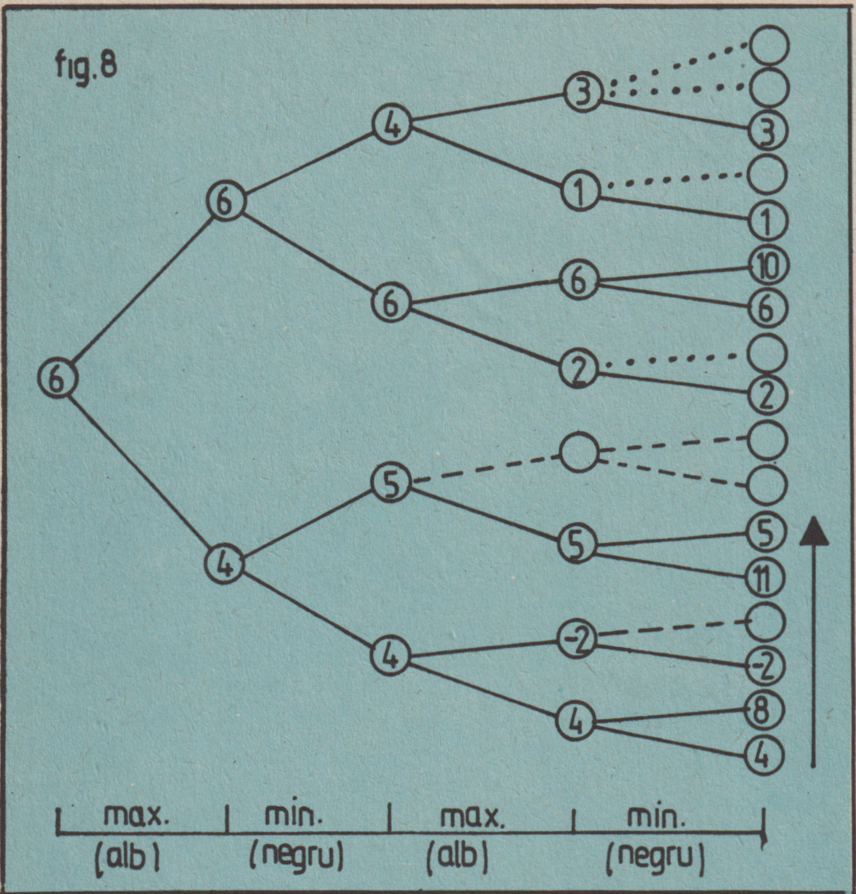
Pentru un nivel k, intervalul a_{k1}, b_{k1} se numește **fereastră.** Este posibil cazul cînd $a_{k1} = b_{k1}$, deci:

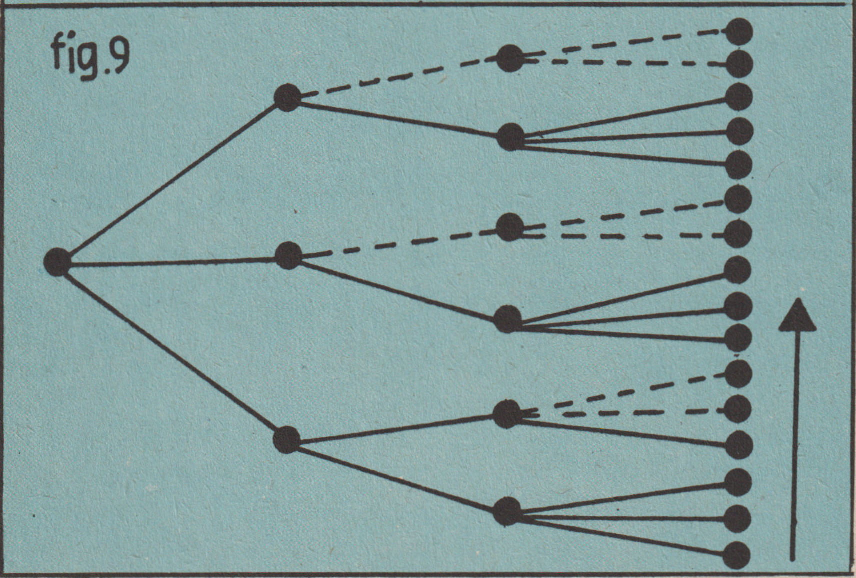
$a_{k1} = a_{k1+1} = \ldots = a_n = b_n = \ldots b_{k1}$

deci **fereastră zero** („zero window"), cum se poate vedea în figura 11.

Am subliniat, în capitolul precedent, că primele variante din arbore, pe fiecare nivel, se „răsfoiesc" mai greu, cît timp valorile mini-max sînt încă cele inițializate cu $\mp\infty$, și cînd criteriul alfa-beta-deep încă nu poate funcționa. Acest lucru face ca algoritmul să-și păstreze un pronunțat caracter combinatorial, chiar cînd se face o foarte bună ordonare a mutărilor în liste.

Oare am folosit toate resursele teoretice pentru a depăși acest impas? Cum putem să abandonăm niște ramuri, cînd criteriul alfa-beta-deep nu funcționează? Trebuie să observăm, totuși, că modul în care se procedează de obicei, într-un algoritm, duce la precauțiile cele mai mari ca soluția să nu fie omisă, ceea ce face ca inițializarea unei valori mini-max să se facă cu $P = \infty$. Adică se merge pe ideea că, oricînd, un jucător poate cîștiga oricît de mult și, de asemenea, în orice moment, adversarul





poate să-i facă un rău oricît de mare. Or, tocmai această teamă, în orice moment, de tot şi de toate, este un punct nerealist al algoritmilor discutaţi mai sus. Dacă s-ar face un calcul al repartiţiei numărului de noduri dintr-un graf ce au o anumită valoare mini-max, s-ar observa că cele mai multe mutări au distribuite valorile lor mini-max în jurul anumitor valori.

Cum se poate oare profita de faptul că valorile mini-max sînt distribuite, cu precădere, în jurul anumitor valori reale? Algoritmul mini-max, în care se aplică criteriul alfa-beta-deep, este un algoritm exact, dar este un algoritm prea prudent.

Putem avea însă o atitudine mai **agresivă** (cînd facem testele conform criteriului alfa-beta-deep), adică, în loc de a iniţializa cu $\mp\infty$ valoarea mini-max pe un nivel, s-o iniţializăm cu o valoare anumită (bine gîndită), s-o notăm cu α (pentru alb) şi cu β (pentru negru). În felul acesta, s-ar putea ca criteriul alfa-beta-deep de reducere de variante **să intre în funcţiune chiar de la prima mutare din listă.** Procedînd astfel, s-ar putea să ajungem în două situaţii:

- există o mutare din listă, pe acel nivel, netăiată de criteriul alfa-beta-deep, **caz bun,** cînd am profitat de încercarea agresivă de a iniţializa forţat valoarea mini-max la o anumită valoare; ea a tăiat ramurile multor mutări dar nu şi pe cea optimă;
- toate mutările din listă au fost tăiate de criteriu, **caz nefavorabil,** cînd analiza trebuie repetată cu α (sau β) iniţializate cu alte valori, sau, în ultimă instanţă, cu $\pm\infty$.

Se înţelege, valorile „agresive" α şi β trebuie alese astfel ca beneficiul tăierilor de ramuri să compenseze cazurile de **reluare** a analizei mutărilor, pe un nivel dat, prin iniţializarea altor valori α şi β.

Impunînd simultan valori α şi β pe două niveluri consecutive, avem de-a face cu o fereastră $[\alpha, \beta]$ în care se încadrează valorile mini-max acceptate. Orice valoare mini-max pe nivelul negrului mai mică ca α va face ca ramurile adiacente acelei mutări să fie abandonate şi face ca orice mutare pe nivelul albului cu valoarea mini-max mai mare ca β să ducă la abandonarea analizei mutărilor adiacente (din aceeaşi listă).

Un caz deosebit (culmea îndrăznelii) este cînd $\alpha = \beta$, deci cînd fereastra $[\alpha, -\beta]$ este redusă la fereastra zero (lărgime zero, „zero window"). **Este cazul cînd valoarea mini-max este cunoscută** (de exemplu se ştie că există mat în 3 mutări) şi se efectuează algoritmul mini-max doar pentru a afla mutarea optimă şi varianta optimă de joc.

Alegerea de intervale $[\alpha, \beta]$ şi reiterarea algoritmului de căutare a mutării optime pot constitui suportul a numeroase experimente pe care le puteţi încerca.

Dacă mulţimea valorilor pe care le poate lua mini-maximul este mică, s-ar putea face reluări ale căutării mutării optime pe rînd, luînd ferestre nule $[\alpha, \beta]$ cu $\alpha = \beta$, pentru toate valorile lui α. Nu vă sfătuim să procedaţi chiar aşa.

Valoarea mini-max P_k pe un nivel **k** se poate afla în următoarele stări:

- iniţializată, neconfirmată de o mutare aleasă;
- confirmată, adică s-a găsit o mutare pe acel nivel cu valoare mini-max egală cu valoarea P_k impusă;
- îmbunătăţită, deci s-a găsit o mutare a cărei valoare mini-max este strict mai bună decît cea P_k, impusă la iniţializare.

Cînd fereastra este nulă, evident, se urmăreşte doar găsirea variantei care să confirme valoarea mini-max impusă.

Ca o remarcă generală, valoarea mini-max pe un nivel poate fi luată în considerare pentru nivelul următor, doar dacă este confirmată, adică valoarea mini-max impusă este egalată sau îmbunătăţită de una din mutările analizate pe acel nivel.

Sintetizînd cele expuse mai sus, putem formula următoarele concluzii:

— dacă P este o valoare mini-max într-un arbore, atunci aplicînd algoritmul alfa-beta-deep-window cu fereastra $[\alpha, \beta]$, cu $\alpha = \beta = P$, se pot determina mutarea optimă şi varianta optimă de joc care realizează acea valoare mini-max;

— dacă $[\alpha_1, \beta_1]$ şi $[\alpha_2, \beta_2]$ sînt două ferestre disjuncte, atunci doar într-una din ele se poate găsi o mutare care confirmă.

Se deduce faptul că putem căuta soluţia încercînd, pe rînd, să aplicăm algoritmul pentru diferite intervale pînă cînd găsim o mutare care confirmă. E o „artă" în a alege cît mai bine aceste intervale!

Toleranţa în aprecierea ponderilor mutărilor. Algoritmul **mini-max** operează foarte exact, dar în mulţimea ponderilor statice şi a valorilor mini-max. Ponderile statice sînt însă aproximative, uneori destul de grosolane, obţinute ca o sumă a tot felul de criterii simplificatoare care se străduiesc să descrie o situaţie.

Să presupunem că, pe un nivel, pentru a putea aplica criteriul alfa-beta, ar mai trebui o foarte mică diferenţă la valoarea curentă. Ştiind că însăşi stabilirea iniţială a ponderilor statice este afectată de erori cu mult mai mari, atunci putem „falsifica" liniştiţi valoarea curentă, ca să favorizăm aplicarea criteriul de reducere. Această falsificare a valorii curente se admite însă cu

cel mult ± T (T este pragul de toleranță), cu scopul aplicării criteriului de reducere. Nu vom aplica însă corecții cînd pe nivelul respectiv încă nu este confirmată nici o mutare. De asemenea, o valoare P_k, care aparține intervalului [α, β], poate fi corectată cu o valoare ±x, însă în așa fel încît: P_k ±x să aparțină intervalului [α, β]. De asemenea, valorile ±x trebuie să nu fie prea mari, ca să nu producă acumulări de erori, aplicîndu-le succesiv pe multe niveluri.

Toleranța face ca algoritmul să nu fie prea „pedant" cînd operează cu valori ce se presupune că sînt calculate imprecis. Totuși, chiar și acest algoritm, aplicînd toleranțe (neînsemnate, e drept), rămîne un algoritm exact.

O remarcă importantă putem face: dacă este găsită o mutare care confirmă într-un interval [α, β] și dacă β-α este mai mic decît toleranța T, atunci putem încheia algoritmul, păstrînd mutarea confirmată ca mutare optimă parțială de pînă atunci.

Toleranța **nu trebuie** aplicată valorilor care descriu **maturi** sau **remize**, căci acestea sînt foarte precis calculate.

Laitmotive (teoria „killerilor").

Presupunem că albul este la mutare și că are regina în priză. În acest caz, la majoritatea mutărilor albului, cea mai bună mutare a negrului este să ia regina. Deci mutarea de luare a reginei figurează ca **laitmotiv**.

În procesul analizei arborelui, dacă pentru fiecare mutare a albului (care nu duce regina din priză) negrul va începe analiza cu mutarea care ia regina, această mutare va face ca majoritatea mutărilor pe nivelurile negrului să fie tăiate, motiv pentru care mutarea de luare a reginei se numește **laitmotiv** (**killer** în limba engleză, adică mutarea care **taie** ramuri). Pentru o mutare a albului, putem aborda ca răspuns al negrului întîi una (sau mai multe) mutări **laitmotiv**.

Pentru a fi eficientă, folosirea mutărilor laitmotiv trebuie să respecte anumite cerințe, atît în privința alegerii mutărilor laitmotiv, cît și a modului de folosire a lor. Nu e prea bine să fie luată ca laitmotiv o mutare care ia piesa care tocmai a mutat. De asemenea, mutarea laitmotiv nu trebuie încercată ca răspuns, cînd ea mută tocmai în locul de unde a plecat o piesă. Oricum, într-o măsură mai mare sau mai mică, mutările laitmotiv trebuie puse în concurență cu mutările evident bune. Mutările laitmotiv pot fi menținute pe orice nivel, totuși ele sînt mult mai necesare spre baza arborelui mutărilor. O mutare laitmotiv nu poate fi reținută decît atunci cînd confirmă (sau îmbunătățește) o valoare mini-max pe un nivel.

Mutările laitmotiv se încearcă în speranța că criteriul alfa-beta va duce la o tăiere a tuturor mutărilor din lista pe acel nivel. Dacă nu reușește, atunci se studiază mutările conform ordonării lor, folosind ponderile statice. Trebuie bine apreciat dacă mutările laitmotiv trebuie să fie încercate sau nu, căci utilizîndu-le abuziv, s-ar putea să creștem factorul de ramificare, în loc să-l micșorăm.

Trebuie să mai remarcăm un fapt interesant, că rărirea arborelui, datorată criteriului alfa-beta, intră în contradicție cu posibilitatea alegerii mutărilor celor mai potrivite ca laitmotiv, tocmai din cauză că arborele are mari porțiuni neanalizate (datorită reducerilor). Mutările laitmotiv au un mare efect de accelerare a analizei arborelui totuși, de aceea, teoria acestora trebuie dezvoltată cu mare acuratețe.

Algoritmi neexacți în analiza arborelui mutărilor.

În programele ASTRO-64 și LABIRINT-64 se proceda astfel: pe fiecare nivel din listele generate, se alegeau cîteva mutări, dintre cele care aveau ponderea statică cea mai bună, și doar pentru acestea se generau mutările succesoare. E drept, arborele era mai suplu, însă mutările abandonate fără analiză creau numeroase neajunsuri, făcînd ca mutările bune să fie omise adesea, pe diferite niveluri. De asemenea, nu erau „văzute" mutări elementare de apărare. Pînă la urmă, în listele reținute, au fost introduse forțat toate mutările care dau șah, ceea ce a dus la rezolvarea unei importante clase de anomalii.

Mai corect este ca anularea unor mutări dintr-o listă completă să se facă doar atunci cînd continuarea nu mai are sens (a funcționat un criteriu de reducere) sau cînd mutarea deja analizată este **mulțumitoare, chiar dacă nu e optimă.**

O schemă interesantă este implementată în programul SUPER-CHESS 3.0, rulat pe calculatoarele SINCLAIR, care se pare că pe nivelurile din arbore în care programul e la mutare, ia în considerare doar anumite mutări („mutări omenești" „human moves"), pe cînd, pe nivelul adversarului, se iau toate mutările. Jocul lui SUPER-CHESS 3.0 este destul de bun (pentru un program de șah) deși, uneori, se mai observă anomalii.

N-am mai pus în evidență faptul că noțiunea de algoritm exact se referă la partea din arbore cu lungimea obligatorie, și nu la prelungirile de ramuri, deși criteriile alfa-beta trebuie aplicate în orice nod al arborelui.

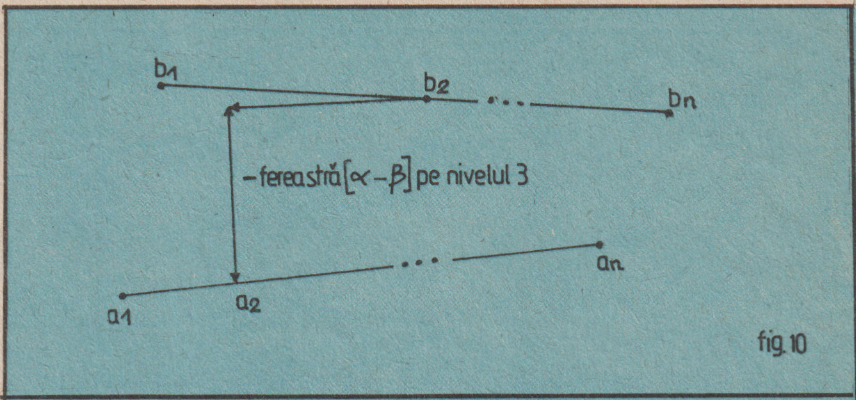


fig.10

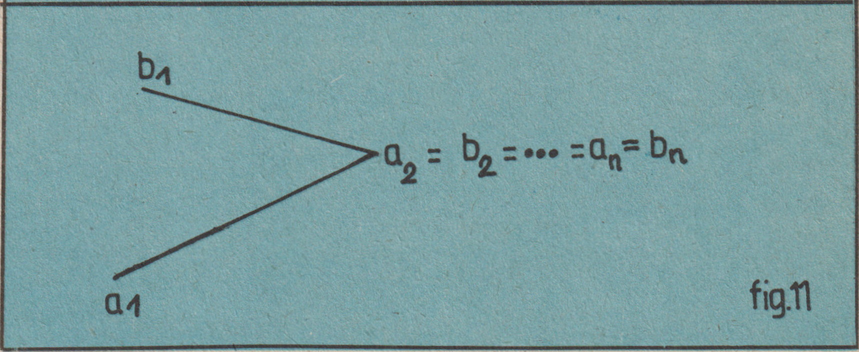


fig.11

ŞAPTE SECRETE...

Traducere şi adaptare după revista RUN COMMODORE ,ian.-febr. 1991

Cu ajutorul a 7 mici tips-uri se poate reuşi mărirea acestei viteze, indiferent de tipul de calculator folosit. O singură aplicaţie poate salva numai o secundă, dar aceste secunde se adună, se adună...

1. - Plasaţi frecvent subrutinele utilizate la începutul programului.

Pentru a înţelege de ce este necesar acest lucru, să cunoaştem cum funcţionează comenzile GOTO şi GOSUB. Cînd se întîlnesc aceste instrucţiuni, BASIC-ul compară linia curentă cu numărul liniei unde se face saltul. Dacă cel de-al doilea număr este mai mare, BASIC-ul începe să caute în continuarea listingului pînă ajunge la această linie. Dacă nu, face un salt la începutul programului, căutînd linia dorită. Majoritatea programelor folosesc această facilitate.

2. - Cînd utilizaţi matrice mari, iniţializaţi toate variabilele neincluse în aceste matrice la începutul programului.

Probabil că aţi obsercat că la iniţializarea matricelor mari, calculatorul „stă" cîteva secunde. Dacă după această iniţializare BASIC-ul găseşte un şir de variabile de iniţializat după aceste matrice, el trebuie să mute tabela matricei în jos, înlocuind-o cu cea a variabilelor nou iniţializate.

Puteţi elimina acest neajuns prin iniţializarea variabilelor (cu DIM spre exemplu), fără a mai da dimensiunea lor:

DIM AB, CD%, EF$

aici BASIC-ul creează trei variabile, setîndu-le la zero (la fel şi în cazul şirului de caractere).

3. - Puneţi numai instrucţiunile absolut necesare în bucle de ciclare.

Puteţi realiza o mare economie de timp „curăţînd" aceste bucle (FOR...NXT;DO...UNTIL). De exemplu, un text nu trebuie să fie afişat decît o singură dată, deci poate fi pus în afara buclei (să zicem, un scor într-un joc).

4. - Folosiţi variabilele date prin litere.

O variabilă ca AD sau EQ% este mult mai rapid citită de BASIC decît 42922. De ce? Pentru că numărul este citit şi convertit într-un format ce poate fi utilizat de calculator, pe cînd o variabilă este citită în mod automat din memoria calculatorului. Se poate verifica uşor să la 2 000 de citiri economia de timp este de aproximativ 30 secunde.

5. - Folosiţi instrucţiunea REM cît mai puţin.

Cînd BASIC-ul întîlneşte această instrucţiune, el trebuie să o recunoască şi apoi să sară la linia următoare, ceea ce încetineşte rularea programului. Normal că această instrucţiune vă ajută la depanarea listingului, dar puteţi să faceţi două programe: unul cu cît de multe REM-uri doriţi, iar al doilea în care eliminaţi această instrucţiune.

6. - Înlocuiţi zerourile (solitare) cu virgula.

BASIC-ul interpretează zero şi virgula în acelaşi fel, dar pe cea de-a doua o recunoaşte mult mai repede. De exemplu:

POKE 53280,0

se poate înlocui cu:

POKE 53280,.

(dar nu POKE 5328.,.)

7. - Nu folosiţi coduri nenecesare.

Multe programe conţin coduri ce nu sînt necesare. Instrucţiunea INT este un exemplu:

POKE 49152, INT(A)

FINAL

Aţi folosit aceste „tips and tricks"-uri şi totuşi programul este încet. Nu vă faceţi probleme, mai sînt alte soluţii:

LIMBAJUL MAŞINĂ — nu este nevoie să rescrieţi întregul program în acest limbaj. Puteţi scrie doar rutinele cele mai importante sau mai des folosite în limbaj maşină.

COMPLETAREA PROGRAMELOR — BASIC-ul de Commodore este doar un interpretor. Se poate compila întregul program în limbaj maşină cu Basic 64 sau Basic 128 sau cu oricare alt compilator, rezultatul fiind mărirea vitezei de lucru de aproape 35 de ori, dar şi ocuparea unui volum de memorie mai mare.

ALGORITMI RAPIZI — analizaţi rutinele pentru a vedea dacă există căi de eliminare a „timpilor morţi".

CĂLIN OBRETIN

DIALOG CU CITITORII

STIMATĂ REDACŢIE,

Numele meu este Mirel Dobrilă, am 31 de ani şi sînt de profesie programator, absolvent al Facultăţii de Cibernetică Bucureşti. Sînt un mare pasionat al informaticii şi doresc să propun cititorilor revistei INFOCLUB o serie de programe scrise în GW-BASIC şi TURBO PASCAL.

Iată cîteva titluri: LABIRINT, GRAPHER, TEST DE CUNOŞTINŢE, CONVERT (unităţi de măsură), AGENDĂ TELEFONICĂ, BOMBER, JOCUL LITERELOR, JUKE-BOX, ANAGRAME, HOROSCOP, GRAFICE MATEMATICE, SIMULATOR AUTO, CODUL MORSE, SUBAH, TEST PSIHOLOGIC etc.

Pentru acest număr vă propun un simulator de maşină electrică de scris, numit "TYPER".

Acest program este scris în Turbo Pascal V.5. şi, evident, poate fi folosit doar cu ajutorul unei imprimante ataşate calculatorului. După introducerea textului sursă, acesta se compilează pentru a elimina eventualele erori de tastare, apoi se poate salva pe disc cu numele dorit (eventual "TYPER"). Apoi se lansează în execuţie cu opţiunea "RUN" a compilatorului, începînd folosirea efectivă a simulatorului.

Dacă se doreşte transformarea programului într-o formă direct executabilă, sub MS-DOS, se poate folosi de pildă compilatorul "TPC.EXE" al firmei BORLAND. Pentru aceasta se încheie sesiunea Turbo Pascal V.5., urmînd introducerea instrucţiunii MS-DOS:

> TPC/E [d:\path\] numefişier [.PAS]

Aici, numefişier este numele sub care s-a salvat anterior programul "TYPER". Ca urmare a comenzii precedente, va rezulta un fişier, "numefişier.EXE", care se poate folosi ca atare, de sub MS-DOS prin simpla invocare a numelui său.

> [d:\path\]numefişier

Ca element inedit în textul sursă al programului "TYPER" se poate remarca metoda de încărcare temporară a unui fişier de caractere introduse de la tastatură.

Acest fişier asignat la perifericul "CON" — consolă — poate conţine la un moment dat maximum 80 de caractere, care sînt de fapt elementele vectorului "strb". Atît timp cît nu se apasă <ESC>, care încheie programul, sau <ENTER>, care "descarcă" şirul de caractere introduse pe un rînd, către imprimantă, vectorul "strb" se "încarcă", aşa cum am menţionat, cu textul introdus. Textul din linia curentă poate fi corectat înaintea apăsării tastei <ENTER>.

"TYPER" apelează cîteva proceduri aflate în "CRT Interface Unit", şi anume ClrScr, Delay, TextColor, TextBackground, Readkey.

MIREL DOBRILĂ

```
program TYPER: { < SYSTEM — SOFT >,
                 BUCUREŞTI, DO-
                 BRILĂ MIREL —
                 1991 simulator de
                 maşină de scris. NE-
                 CESITĂ PRINTER!
                 textul se poate edita cu
                 cursorul şi cu Back-
                 space; şirul de carac-
                 tere este trimis spre
                 printer cu < CR > }
uses Crt; { se folosesc: TextColor, Text-
            Background, ClrScr, Delay,
            Readkey }
var filename : string[80]; { rîndul poate
                             avea max. 80
                             caractere }
  f, lst      : text
  strb        : string
  procedure init;           { pregătire ecran }
  begin
    ClrScr;
    TextBackground (10); TextColor (0);
    write ('MAŞINA DE SCRIS ELEC-
           TRICĂ: Cursor, Delete, Backspace,
           <Enter>=Print');
    write (', <Esc>=Stop ');
    TextBackground (0); TextColor (7);
    writeln; writeln;
  end;                                   { init }
procedure putchr (strb: string); { încarcă
                         vectorul cu caractere }
var i      : integer ;
    le     : integer ;
    ch     : char    ;
begin
  i   := 1;
  le := length (strb);
  while i <= le do
  begin
    write (lst, strb[i]);
    inc(i);
  end;
  writeln (lst);                        { rînd nou }
end;                                    { putchr }
begin                           { program principal }
  init;
  assign (lst,' lpt1'); rewrite(lst);
  if paramcount = 1 then
    filename : = paramstr (1)
  else
    filename := 'CON';
assign ( f, filename);
reset (f);
writeln;
while Readkey < > # 27 do          { testează
                                   apasare Esc }
  begin
  write (#26);
    readln (f, strb);
    putchr (strb);
    end;
  close (f);
    ClrScr; writeln; writeln; writeln;
    Writeln('                PE CURÎND ! ');
    Delay (1000);
    ClrScr;
end.                             { sfîrşit program }
```

Florin VASILIȚĂ

PROGRAM PENTRU ANALIZA ȘI REORGANIZAREA SPAȚIILOR ÎNTR-UN TEXT

Pe ecranul TV al unui calculator compatibil Sinclair Spectrum într-un rînd de text intră un număr fix de caractere. În acest caz deseori se întîmplă ca ultimul cuvînt de pe rînd să fie trunchiat. Programul analizează cîte cuvinte pot intra într-un rînd dat, iar cuvîntul care ar urma să fie trunchiat se trece pe rîndul următor. Spațiile dintre cuvinte se modifică în așa fel încît ultimul cuvînt rămas să se termine la sfîrșitul rîndului (adică aliniat la dreapta). Acest tip de program subrutină se poate utiliza în orice program de editare și prelucrare de texte, dar și în programe de aplicații care utilizează texte.

ALGORITMUL: Se analizează ultimul caracter din rînd. Dacă acesta este litera sau orice alt caracter, cu excepția spațiului, atunci sîntem în cazul unui cuvînt și se va analiza în continuare următorul caracter. Dacă acesta este spațiu, e bine, înseamnă că sîntem în cazul în care sfîrșitul unui cuvînt se termină la sfîrșitul rîndului. În toate celelalte cazuri vom proceda în așa fel încît ultima literă din ultimul cuvînt să se termine la sfîrșitul rîndului, adică textul să fie aliniat la dreapta.

Dacă următorul caracter nu este spațiu, atunci sîntem în interiorul unui cuvînt care va trebui trecut pe rîndul următor. Este necesară introducerea de spații între cuvintele rămase pînă cînd ultima literă din ultimul cuvînt este aliniată la dreapta. Se analizează spre stînga caracter cu caracter și se numără caracterele. Cînd se întîlnește caracterul spațiu, se pune și el „la număr" și se oprește analiza. Numărul de caractere (plus spațiul găsit) trebuie împărțit la spațiile rămase pe rînd, distanța dintre cuvinte mărindu-se cu cîtul acestei împărțiri.

Dacă împărțirea nu este exactă, atunci se distribuie spațiilor dintre cuvinte, începînd de la spațiul dinaintea ultimului cuvînt, cîte un spațiu din rest. În cazul altor algoritmi, se poate pune restul începînd cu spațiul de după primul cuvînt. Noi am ales varianta de punere a restului înaintea ultimului cuvînt.

Revenind la începutul analizei, am descris cazul în care ultimul caracter din rînd era un caracter diferit de spațiu.

În cazul în care ultimul caracter este spațiu, acest spațiu va fi pus alături de spațiul dinaintea ultimului cuvînt, prin deplasarea ultimului cuvînt la dreapta cu un spațiu.

Programul va trebui să ignore de asemenea spațiile dinaintea primului cuvînt pentru ca textul să fie aliniat și la stînga. În acest scop, spațiul dinaintea primului cuvînt va fi pus la dreapta ultimului cuvînt din rîndul anterior.

DESCRIEREA PROGRAMULUI: Pentru a putea fi folosit ca o subrutină, liniile programului au fost numerotate începînd cu 9100.

```
9100> DIM B(B28)
9102 LET J=1
9104 FOR K=1 TO B28
9106 IF G$(K TO K)=" " THEN  LET
B(J)=K: LET J=J+1
9108 NEXT K
9110 LET NJ=J-1
9112 IF B(NJ)=B28 THEN  GO TO 914
6
9114 LET SPA=B28-B(NJ)
9116 LET RE= INT (SPA/NJ)
9118 REM
9120 IF B(NJ)=B28-1 THEN  LET NJ=
NJ-1
9122 LET NJ1=NJ-1: IF NJ <= 1 THE
N  LET NJ1=1
9123 FOR J=NJ1 TO 1 STEP -1
9124 LET H$=G$(1 TO B(J))+"  "+G$(
B(J)+1 TO )
9126 LET G$=H$
9128 LET B(J)=B(J)+1
9130 IF NJ>1 THEN  FOR M=J+1 TO N
J-1: LET B(M)=B(M)+1: NEXT M
9132 LET SPA=SPA-1
9134 IF SPA=0 THEN  GO TO 9146
9136 NEXT J
9138 IF SPA>0 THEN  GO TO 9122
9140 LET RE=RE-1
9142 IF RE>0 THEN  GO TO 9122
9146 RETURN
9147
9148 REM
9149
9150 LET B28=B28-TA
9151 LET G$=E$(1 TO B28)
9152 GO SUB 9100
9154 PRINT  TAB TA:G$(1 TO B28)
9156 LET F$=G$+E$(B28+1 TO )
9158
9162 LET F$=F$(B28+1 TO )
9164 LET LA= LEN F$
9166 LET R= INT ((LA-B28)/B28+.97
)
9168 IF R<1 THEN  GO TO 9184
9170 LET G$=F$(1 TO B28)
9172 GO SUB 9100
9174 PRINT  TAB TA:G$(1 TO B28)
9178 LET F$=G$+F$(B28+1 TO )
9182 GO TO 9162
9184 PRINT  TAB TA:F$
9186 RETURN
9187
9188
9189
```

Comentarii

Rutina principală se află între liniile 9150 și 9186. Numărul de caractere de pe o linie este memorat în variabila B28. Rutina ia cîte B28 caractere și le transmite rutinei de expandare care începe la linia 9100. Aceasta este între liniile 9150 și 9152. Rutina de expandare este apelată în linia 9152:

9152 GO SUB 9100.

Șirul G$ este tipărit de la caracterul 1 pînă la caracterul B28 (linia 9154), după care caracterele cuvîntului care n-au încăput pe rînd sînt trecute în șirul F$ (liniile 9156—9162). Procesul se repetă: se separă cîte un rînd de B28 de caractere și este atribuit lui G$, fiind apoi trimis rutinei de expandare de la 9100 (liniile 9170—9182). Dacă rămîne un rest mai mic decît B28, atunci acesta este tipărit fără a fi expandat ca sfîrșit de paragraf (linia 9184).

Între liniile 9100 și 9146 se află rutina care realizează expandarea unui rînd G$ prin inserarea de spații între cuvinte și deplasarea cuvîntului (care altfel ar trebui trunchiat) la dreapta în afara celor B28 de caractere tipăribile. Rutina de expandare are ca parametri lungimea rîndului (B28) și șirul de text (G$).

Liniile 9100—9110 realizează un tablou B(J)=K care conține adresele spațiilor dintr-un rînd.

Dacă adresa ultimului spațiu este tocmai sfîrșitul rîndului, se iese din rutina de expandare (linia 9112).

În linia 9114 se atribuie lui SPA evaluarea diferenței lungimii rîndului B28 și adresa ultimului spațiu B(NJ), adică numărul de spații care trebuie redistribuite la celelalte intervale dintre cuvinte. În linia 9116 se atribuie variabilei RE rezultatul împărțirii lui SPA la NJ (numărul de intervale dintre cuvinte).

Între liniile 9122 și 9146 se realizeaza redistribuirea spațiilor RE intervalelor dintre cuvintele rămase pe rîndul analizat.

MODUL DE UTILIZARE:

Înainte de apelul rutinei principale trebuie date valori pentru parametri: TA (marginea din stînga a textului), B28 (marginea din dreapta) și E$ (textul propriu-zis), după care se face apelul subrutinei cu instrucțiunea:

GO SUB 9150.

Se alcătuiește astfel o mică subrutină care începe la linia 1000 și care conține parametrii și apelul subrutinei.

Exemplu:

```
  10 LET E$="PE CULMEA CEA MAI IN
ALTA A MUNTILOR CARPATI SE INTIND
E O TARA MINDRA SI BINECUVINTATA
INTRE TOATE TARILE DE PRE PAMINT
..."
  20 GO SUB 1000
 999 STOP
1000 LET TA=4: LET B28=30: GO SUB
 9150: RETURN
```

Textul E$ (linia 10) definește textul ce va fi tipărit începînd de la TA=4 marginea din stînga, pînă la B28=28 marginea din dreapta (linia 1000 definește o subrutină de apel a subrutinei principale). Textul tipărit de la o margine la alta se poate observa în figura următoare:

```
PE CULMEA CEA  MAI INALTA
A  MUNTILOR  CARPATI  SE
INTINDE O TARA  MINDRA SI
BINECUVINTATA INTRE TOATE
TARILE DE PRE PAMINT ...
```

În cazul că dorim să schimbăm încadrarea textului, de exemplu între coloanele 10 și 30, în instrucțiunea de la linia 1000 modificăm TA și B28 astfel:

```
1000> LET TA=10: LET B28=30: GO S
UB 9150: RETURN
```

Rezultatul va fi:

```
PE CULMEA  CEA  MAI
INALTA  A  MUNTILOR
CARPATI  SE INTINDE
O  TARA  MINDRA  SI
BINECUVINTATA INTRE
TOATE TARILE DE PRE
PAMINT ...
```

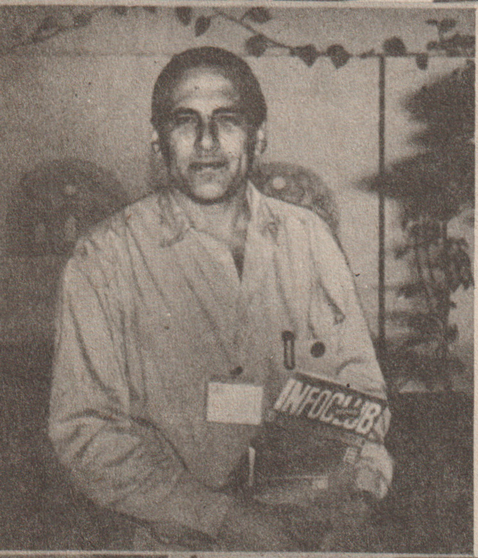
VARIABILE ȘI TABLOURI FOLOSITE: Variabilele folosite de rutine, deci care ar putea fi alterate în cazul înglobării subrutinei într-un alt program, sînt: E$, G$, F$, B(B28), LA, R, J, K, NJ, SPA, RE, M. Ele au fost puse în ordinea în care apar în subrutina principală și apoi în subrutina secundară.

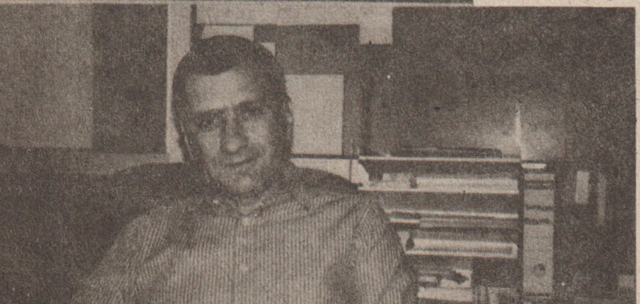
CONCLUZII

Dacă textul prezintă mai multe spații între cuvinte, acestea nu mai sînt compactate, rămînînd deci loc pentru alte dezvoltări ale programului. Am considerat că modul natural de introducere a unui text este cu un spațiu între cuvinte. Dacă este vorba de un text spațiat care trebuie rearanjat, mai întîi trebuie reduse spațiile dintre cuvinte la unul singur și apoi apelată rutina care rearanjează cuvintele în rînduri. De asemenea rutina de expandare poate fi îmbunătățită astfel ca, pentru orice text dat, să existe posibilitatea expandării între anumite limite date. Aceste rutine pot fi dezvoltate astfel încît să acopere o gamă de probleme în funcție de anumite cerințe de tipărire. În cazul nostru, am avut cerința de a tipări un text aliniat la stînga și la dreapta de anumite limite impuse și care prezenta cîte un singur spațiu între cuvinte.









Un succes de prestigiu
al şcolii româneşti de informatică!

OLIMPIADA INTERNAŢIONALĂ DE INFORMATICĂ — ATENA 1991

Fireşte că o olimpiadă şi, mai ales, una internaţională poate fi abordată din multiple unghiuri. Sarcina devine însă mult mai dificilă în momentul în care olimpiada este de informatică şi a avut loc în Grecia. Informatică şi istorie milenară, într-un loc de vis şi într-o interesantă simbioză. Aşadar, a treia ediţie a Olimpiadei internaţionale de informatică (la care a participat şi subsemnata în calitate de... „observator"!) a aliniat la start reprezentanţi din 23 de ţări de pe toate continentele. Delegaţia noastră a avut drept reprezentanţi în concurs pe **Cătălin VOINESCU** (Braşov), **Mihai MANOLESCU** (Bucureşti) şi **Radu PREDA** (Bacău) conduşi de dr. mat. **Stelian NICULESCU**, inspector general cu probleme de informatică în Ministerul Învăţămîntului şi Ştiinţei, şi conf. univ. **Horia GEORGESCU**, Universitatea din Bucureşti, Facultatea de Matematică. La capătul unui concurs dificil, cu concurenţi „serioşi" care s-au dovedit adversari redutabili şi la ediţiile anterioare (Pravetz şi Minsk), elevii români au repurtat un succes deosebit: cîte un premiu trei fiecare! Desigur că emoţiile au fost mari, probele (mai ales cea din a doua zi de concurs) dificile, iar concurenţii noştri au dorit, fireşte, un premiu 1 sau 2. Important pentru noi este că numele României a fost auzit de trei ori la festivitatea de premiere, la care au participat numeroase oficialităţi şi personalităţi ale vieţii politice şi ştiinţifice greceşti, reprezentanţi ai unor publicaţii de specialitate (cum ar fi „Software & Computers", unul dintre sponsorii manifestării), firme de profil care au oferit şi premii etc. Se pot scrie multe despre acest gen de manifestări care reunesc oameni atît de diferiţi din toate punctele de vedere şi se pot lua multe interviuri (există în aceste cazuri un material imens pentru un ziarist!), dar dintre toate am selectat o discuţie pe care am avut-o în timpul concursului cu domnul **CHRISTOS KILIAS**, profesor asociat de informatică la „Technological Educational Institute" din Atena şi, pe parcursul olimpiadei, preşedintele comitetului de organizare.

— Stimate domnule Kilias, o să încep cu o întrebare obişnuită în astfel de ocazii: care este părerea dumneavoastră despre participarea la această ediţie a olimpiadei?

— Da, mă aşteptam la această întrebare. După cum cred că ştiţi deja, participă efectiv 23 de ţări, 3 state au trimis observatori şi, fireşte, UNESCO are, de asemenea, un observator. Ne aşteptam la o participare mai mare, dar ea nu s-a putut realiza datorită unor condiţii obiective de ambele părţi; anul şcolar nu este încă încheiat, în unele ţări nu s-au terminat competiţiile naţionale, apoi, situaţia internaţională cu multe probleme, iar de partea noastră faptul că nu am putut amîna data concursului la începutul lunii septembrie, cum a fost iniţial planificată, datorită imposibilităţii obţinerii în acea perioadă a acestui hotel-şcoală cu spaţiile respective de cazare şi clase, deci adecvat scopului nostru.

— Care este nivelul competiţiei din acest an?

— Foarte ridicat! După proba din prima zi, de nivel mediu de dificultate, proba de azi (ziua finală a concursului – n.r.) este foarte dificilă. Concurenţii au la dispoziţie calculatoare personale compatibile IMB PC, puse la dispoziţie pe timpul concursului de unul dintre sponsorii noştri. De altfel, pot să apreciez probele ca fiind de nivel universitar.

— Puteţi face un pronostic asupra posibililor învingători?

– Îmi puneţi o întrebare dificilă şi la care îmi este foarte greu să răspund. Ceea ce vă pot spune este că la celelalte ediţii echipele URSS, Bulgaria, China şi chiar a dumneavoastră au făcut o impresie foarte bună (în mare parte pronosticul d-lui Kilias a fost corect, cu unele surprize plăcute, între care faptul că primul clasat a fost Igor Maly din Cehoslovacia, urmat la mică distanţă de reprezentanţii Chinei, Ungariei, Iugoslaviei, Bulgariei, Poloniei, URSS etc. – n.r.), dar, oricum, o competiţie ca aceasta nu exclude surprize plăcute pentru unii, iar pentru alţii... Ceea ce aş mai vrea să spun este că sînt foarte impresionat de prezenţa presei din România aici (a fost şi o delegaţie a Televiziunii Române – n.r.), ceea ce demonstrează multe, multe lucruri şi toate, fireşte, pozitive.

I-am mulţumit domnului Kilias pentru amabilitate, discuţia a fost de fapt mult mai amplă, inclunzînd şi alte probleme legate de sistemul de învăţămînt în Grecia, de perspectivele informaticii, de modul în care gazdele au reuşit să organizeze o astfel de manifestare.

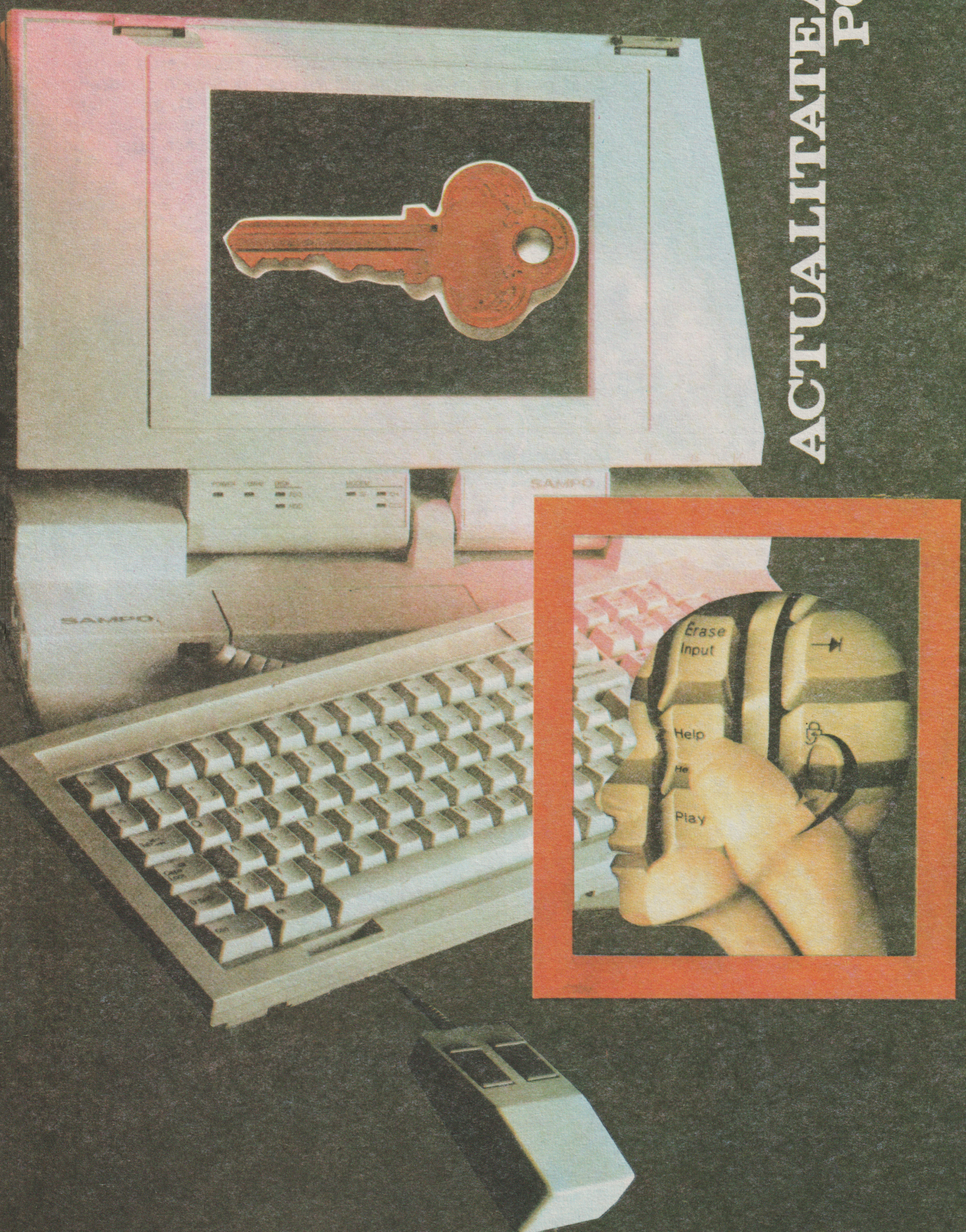
Ar mai fi multe de spus. De pildă faptul că între cei 27 de laureaţi s-a aflat şi o fată (Sophia Dekkers din Olanda), că în Grecia există o gamă largă de reviste de informatică, în afară de cele şase ale IDG, că „Greek Computer Society" a fost printre sponsorii activi ai acestei manifestări, pe scurt că preocupările sînt numeroase, multe dintre ele comune cu ale noastre şi că acest schimb de experienţă a fost util şi benefic. Înainte de a încheia, ţin să aduc mulţumiri membrilor Ambasadei României de la Atena pentru sprijinul nemijlocit pe care mi l-au acordat în îndeplinirea cu succes a misiunii mele la Atena.

Felicitări celor trei laureaţi şi sperăm cel puţin acelaşi succes la ediţia a 4-a, care va avea loc la anul la BONN!

Mihaela GORODCOV

1. Delegaţia română învingătoare.
2. Profesorul **E.J.Yannakoudakis** — Athens University of Economics and Business — al cărui interviu va fi publicat în nr. 8/1991 al revistei „Ştiinţă şi tehnică".
3. Profesorul **Christos Kilias**, preşedintele comitetului de organizare.
4. **Chris Samouhos**, president H.A. Samouhos Publications Ltd-IDG Grecia.

ACTUALITATEA PC



Doina ISTRĂTESCU,
Eugen GEORGESCU

CURSOARE GRAFICE

Aşa după cum aţi putut remarca, acest serial a încercat pînă acum să vă familiarizeze cu cuploarele grafice curente ale calculatoarelor IBM PC (PS/2), să vă familiarizeze într-un mod cît mai interactiv cu implementarea primitivelor grafice. Ceea ce a lipsit însă pînă acum a fost chiar... interactivitatea. Obiectul grafic, absolut necesar pentru aceasta, este aşa-numitul cursor grafic care, ataşat unor periferice de intrare, cum ar fi mouse, tabletă grafică, joystick, tastatură, ne va permite să ne integrăm în lumea ecranului.

ALU* are un rol esenţial în realizarea unei forme geometrice care poate fi mutată în timp real pe suprafaţa ecranului, peste zone deja desenate, **fără a le altera.** În operaţia de desenare a oricărui tip de cursor grafic, ALU este programată să funcţioneze în modul XOR, după care, la încheierea operaţiei, este restaurat modul specificat în variabila internă „replacement-rule". Funcţia XOR nu este altceva decît o operaţie de inversare comandată.

Activitatea unui tip de cursor (ECHO-Switch) înseamnă o desenare. Mutarea unui cursor într-o poziţie nouă (ECHO-Move) constă în redesenarea peste vechea poziţie şi în activarea pe noua poziţie. Două desenări peste aceeaşi poziţie echivalează cu două inversări, deci cu revenirea la starea iniţială.

Pe un ecran pot fi active simultan mai multe cursoare, de tipuri diferite sau de acelaşi tip. În cazul în care sînt de acelaşi tip, ele pot diferi prin culoare. Deci pentru fiecare din ele trebuie să memorăm poziţia curentă (x, y), tipul şi culoarea. Numărul de cursoare grafice ce pot fi active simultan este dat de numărul de structuri de date pentru care rezervăm spaţiu în memorie, iar culorile este bine să le predefinim.

Au fost implementate un număr de 5 cursoare grafice clasice:

SMALL-CROSS: derivat din marcajele reticulare ale sistemelor optice, este oarecum clasic. Deşi pe calculatoarele personale nu mai este la modă, el rămîne totuşi cel mai precis din punct de vedere al intercorelării cu alte elemente grafice deja prezentate pe ecran.

WIDE-CROSS: o versiune a primului, constînd dintr-un sistem ortogonal trasat între marginile extreme ale zonei de desenare. Este folosit în special pentru alinieri cu alte elemente grafice aflate la distanţe mai mari pe ecran.

RUBBER-LINE: constă într-o linie între un punct fix — ultima coordonată a unei operaţii line(), draw() sau move() — şi poziţia curentă a locatorului (mouse, tabletă) ataşat. Utilizarea sa principală este trasarea de linii perfect verticale sau orizontale. Acesta este singurul caz în care efectul de aliasing are un rol pozitiv, „ruperile" liniei fiind uşor observabile şi semnalizînd chiar şi cea mai mică (1 pixel) abatere de la direcţia verticală sau orizontală.

RUBBER-RECTANGLE: este un dreptunghi avînd un colţ fix — ultima coordonată a unei operaţii line(), draw() sau move() —, iar colţul diagonal este determinat de poziţia curentă a locatorului (mouse, tabletă). El este utilizat cu precădere pentru încadrarea într-un areal a unui grup de entităţi grafice preexistente în scopul ştergerii, copierii sau mutării lor.

SMALL-ARROW: este o mică săgeată, orientată stînga sus la 45°, avînd punctul curent în vîrf. Din punct de vedere funcţional, este folosit la fel ca primul tip de ecou, dar precizia sa de atingere a unui pixel e mai scăzută. La modă este o versiune mult mai mare, dar tocmai acolo unde este folosită precizia cerută e mai scăzută: selecţia unor meniuri (Ventura, Orcad) sau a unor „butoane" (Windows).

Trecînd acum la sursa C, veţi observa ca numai primele două funcţii sînt pentru utilizator, restul fiind interne bibliotecii. Se remarcă în special două: **draw_v** (trasarea unei linii verticale) şi **draw_h** (trasarea unei linii orizontale). Un ecou trebuie să urmărească în timp real evoluţia unui locator şi deci viteza este un factor prohibitiv. Trei din ecourile prezentate se bazează pe linii verticale şi orizontale. În aceste cazuri particulare se pot imagina algoritmi de 5—10 ori mai rapizi decît cel general (Bresenham).

Trasarea verticală aduce sporul maxim de viteză fiindcă scoate programarea registrului 8 al controller-ului în afara procesului iterativ şi deci se execută o singură dată. Reprogramările registrelor EGA/VGA sînt elementele care duc la cele mai mari consumuri de timp, mai ales că ele nu se găsesc pe motherboard (placa de bază) şi operaţiile respective de CPU trebuie să iasă prin magistrală. Din acest motiv, la multe din calculatoarele mai noi, cuplorul VGA este built-in (inclus) şi cu acces pe 16 biţi astfel încît se pot ataca două registre simultan. Revenind la algoritm, observăm că trasarea unei linii s-a redus la înscrierea periodică (lungimea liniei = 80 bytes = RES_X) a măştii dorite.

Pentru trasarea orizontală situaţia este un pic mai complicată. Pentru înţelegerea algoritmului să analizăm cazul general al liniei orizontale (fig. 1).

După cum se vede, pe cazul general al liniei orizontale se remarcă trei zone:

— ZONA II: linia ocupă în întregime octetul şi se programează o singură dată registrul 8 cu masca 0xff, după care se completează succesiv toţi octeţii zonei.

— ZONA I: începutul liniei activează pixelii de la poziţia curentă pînă la sfîrşitul octetului (dreapta). Pentru a simplifica la minimum calculele pentru mască, acestea se regăsesc pretabelate în variabila right_mask (8).

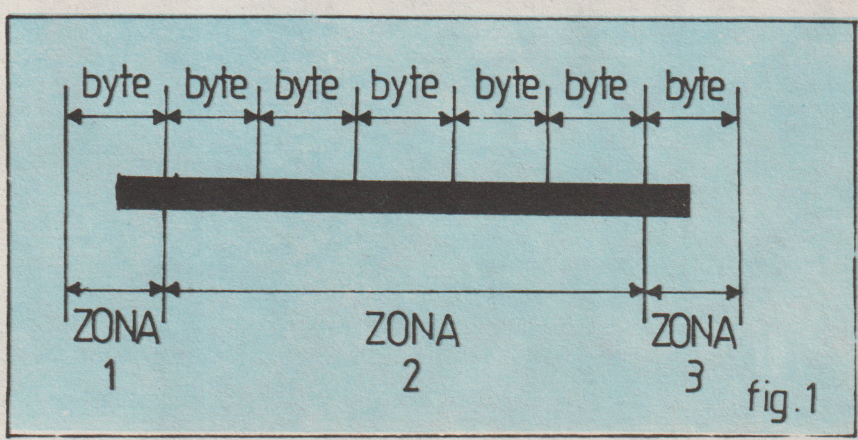


fig.1

LIB.H :

```
struct  ECHO     {
                 unsigned int x;
                 unsigned int y;
                 unsigned char type;
                 unsigned char color;
                 };
```

LIB_DC.H :

```
#define LG_CRS   5                /* Small Cross Echo dimension */
#define ECHO_SW  0                /* ECHO SWitch */
#define ECHO_MV  1                /* ECHO MoVe */
#define RES_X    640/8            /* Nr bytes/line */

extern  char     far      *p_ega;

/* -----------------------------------------------------------------------
/* Zona de variabile necesare bibliotecii */
struct  ECHO     echo0,echo1,echo2,echo3; /* pozitia ecoului anterior*/
struct  ECHO     *pecho;
unsigned int     last_x,last_y;          /* current point */
unsigned char    right_mask[8]={0xff,0x7f,0x3f,0x1f,0x0f,0x07,0x03,0x01};
unsigned char    left_mask[8] ={0x80,0xc0,0xe0,0xf0,0xf8,0xfc,0xfe,0xff};
unsigned char    pxmask[8]    ={0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned int arrow[8]= {0x0f00,0x0700,0x0700,0x0900,0x1000,0x2000,
                         0x4000,0x8000};
/* -----------------------------------------------------------------------

echo_type(echo_number,echo_value,dcx,dcy)
        unsigned int    echo_number,echo_value,dcx,dcy;
{
        if(dcx>638)
                dcx = 638;
        if(dcy >348)
                dcy = 348;
        if(pecho->type)
                echo(pecho->x,pecho->y,pecho->type,ECHO_MV);
        switch(echo_number)
          {     case 0 : pecho = &echo0;
                         break;
                case 1 : pecho = &echo1;
                         break;
                case 2 : pecho = &echo2;
                         break;
                case 3 : pecho = &echo3;
                         break;
          }
        if(pecho->type)
                echo(pecho->x,pecho->y,pecho->type,ECHO_MV);
        if(echo_value)
                echo(dcx,dcy,echo_value,ECHO_SW);
        pecho->type = echo_value;
}
echo_update(echo_number,dcx,dcy)
        unsigned int    echo_number,dcx,dcy;
{
        switch(echo_number)
          {
```

Actualitatea PC

```
        outp(P_ctr_r,0x00);
        outp(P_ctr_a,0x00);
        outp(P_ctr_r,0x00);
}


draw_v(x,begin,end)
        int     x,begin,end;
{
        char    far *pe;
        register int    tmp,count,j;
        if(begin>=end)
                {tmp=begin;begin=end;end=tmp;};
        count = end-begin-1;
        pe =p_ega+begin*80+(x>>3);
        outp(P_ctr_a,8);
        outp(P_ctr_r,0x80 >> (x&7));
        for (j=0 ; j < count ; j++)
                *(pe+=80) |= 0xff;
}

draw_h(y,begin,end)
        int     y,begin,end;
{
        char    far *pb,far *pe;
        register int    i,b8,e8,yN,count;
        if(begin>=end)
                {i=begin,begin=end,end=i;}

        b8=begin/8,e8=end/8,yN=y*N_OCT;
        count = e8-b8;
        pb=p_ega+yN+b8;
        pe=p_ega+yN+e8;
        if(b8==e8)
                { outp(P_ctr_a,8);
                outp(P_ctr_r,right_mask[begin&7] & left_mask[end&7]);
                *pb |= 0xff;
                return; }
        else
                { outp(P_ctr_a,8);
                outp(P_ctr_r,right_mask[begin&7]);
                *pb |= 0xff;
                outp(P_ctr_a,8);
                outp(P_ctr_r,0xff);
                for(i=1 ; i < count ; i ++ )
                        *(pb+i) |= 0xff;
                outp(P_ctr_a,8);
                outp(P_ctr_r,left_mask[end&7]);
                *pe |= 0xff;
                }
}
        case (6) :
                if(action)
                        for(i=0;i<8;i++)
                            { byte = (pecho->x>>3) + RES_X*(pecho->y+i);
                            rx = pecho->x & 7;
                            preg->x.ax = arrow[i]>>rx;
                            outp(P_ctr_a,8);
                            outp(P_ctr_r,preg->h.ah);
                            *(p_ega+byte-1) |= 0xff;
                            outp(P_ctr_a,8);
                            outp(P_ctr_r,preg->h.al);
                            *(p_ega+byte) |= 0xff; }
                for(i=0;i<8;i++)
                        { byte = (xp>>3) + RES_X*(yp+i);
                        rx = xp & 7;
                        preg->x.ax = arrow[i]>>rx;
                        outp(P_ctr_a,8);
                        outp(P_ctr_r,preg->h.ah);
                        *(p_ega+byte-1) |= 0xff;
                        outp(P_ctr_a,8);
                        outp(P_ctr_r,preg->h.al);
                        *(p_ega+byte) |= 0xff; }
                pecho->x=xp , pecho->y=yp;
                break;
```

```
                case 0 : pecho = &echo0;
                         break;
                case 1 : pecho = &echo1;
                         break;
                case 2 : pecho = &echo2;
                         break;
                case 3 : pecho = &echo3;
                         break;
            }
        echo(dcx,dcy,pecho->type,ECHO_MV);
}


echo(xp,yp,echo_type,action)
        int xp,yp,action;
        unsigned char echo_type;
{
        unsigned int byte;
        char i,rx;
        outp(P_ctr_a,3) ;
        outp(P_ctr_r,0x18);
        outp(P_ctr_a,0x01);
        outp(P_ctr_r,0xff);
        outp(P_ctr_a,0x00);
        outp(P_ctr_r,pecho->color);
        outp(P_seq_s,0x02);
        outp(P_seq_d,0x0f);
        switch (echo_type)
        {
        case (1) :
        case (3) :
                if(action)
                     { draw_h(pecho->y,pecho->x-LG_CRS,pecho->x+LG_CRS);
                     draw_v(pecho->x,pecho->y-LG_CRS,pecho->y+LG_CRS); }
                draw_h(yp,xp-LG_CRS,xp+LG_CRS);
                draw_v(xp,yp-LG_CRS,yp+LG_CRS);
                pecho->x=xp , pecho->y=yp;
                break;
        case (2) :
                if(action)
                        { if(xp != pecho->x)
                                { draw_v(pecho->x,0,349);
                                draw_v(xp,0,349);
                                pecho->x = xp; }
                        if(yp != pecho->y)
                                { draw_h(pecho->y,0,639);
                                draw_h(yp,0,639);
                                pecho->y = yp; }
                        }
                else
                        { draw_v(xp,0,349);
                        draw_h(yp,0,639);
                        pecho->x=xp , pecho->y=yp; }
                break;
        case (4) :
                if(action)
                        e_line(last_x,last_y,pecho->x,pecho->y);
                e_line(last_x,last_y,xp,yp);
                pecho->x=xp , pecho->y=yp;
                break;
        case (5) :
                if(action)
                        { draw_v(pecho->x,last_y,pecho->y);
                        draw_v(last_x,pecho->y,last_y);
                        draw_h(last_y,last_x,pecho->x);
                        draw_h(pecho->y,pecho->x,last_x); }
                draw_v(xp,last_y,yp);
                draw_v(last_x,yp,last_y);
                draw_h(last_y,last_x,xp);
                draw_h(yp,xp,last_x);
                pecho->x=xp , pecho->y=yp;
                break;
```

```
        }
        outp(P_ctr_a,3) ;
        outp(P_ctr_r,replacement_rule);
        outp(P_ctr_a,8);
        outp(P_ctr_r,0xff);
        outp(P_seq_s,0x02);
        outp(P_seq_d,write_mask);
        outp(P_ctr_a,0x01);
e_line(xp,yp,xf,yf)
        unsigned        xp,yp,xf,yf;
{
        register int    iv,px,py;
        int     byte,nx,ny,nt,nd,ns;
        unsigned int    anx,any;

        nx = xf - xp;
        ny = yf - yp;
        px = (nx > 0) ? 1 : -1;
        py = (ny > 0) ? 1 : -1;
        anx = abs(nx);
        any = abs(ny);
        if(anx > any)
        {
        nt=anx;
        nd=any;
        }
else
        {
        nt=any;
        nd=anx;
        }
    ns = nt - nd;
    iv = (nt>>1) - ns;
    while(nt)
        {
        byte = (xp>>3)+RES_X*yp;
        outp(P_ctr_a,8);
        outp(P_ctr_r,pxmask[xp & 7]);
        *(p_ega+byte) |= 0xff;
        nt--;
        if( iv >= 0 )
                { iv -= ns;
                xp += px;
                yp += py; }
        else
                { iv += nd;
                if(anx > any)
                        xp += px;
                else
                        yp += py;
                }
        }
```

— ZONA III: este cazul complementar al zonei I, iar măștile pretabelate pentru registrul 8 al controller-ului se găsesc în variabila left_mask (8).

Există și un caz particular cînd începutul și sfîrșitul liniei se găsesc în cadrul aceluiași octet. În acest caz, singura mască utilizată se obține prin intersecția măștilor din cazurile I și III.

Ce facem cu aceste cursoare grafice? Într-o aplicație profesională avem driver atașat locatorului care lucrează în întreruperi și apelează direct funcția **echo_update()**. Aceasta implică un anumit nivel de cunoștințe asupra sistemului de operare MS-DOS și limbajului C, ceea ce ar fi o deviere de la subiectul articolului. Vă propunem o soluție banală și aflată la îndemîna tuturor.

Există un driver clasic, denumit MOUSESYS.COM, care face ca mișcările unui mouse să aibă același efect ca folosirea săgeților de pe tastatura, iar butoanele sale vor acționa ca tastele funcționale F1, F2, (F3). Ne vom reduce astfel la o problemă de tastatură. Caracteristic tuturor acestor taste este faptul că dau cod dublu, adică se generează două caractere din care primul e 0.

Ca să vă puteți bucura de efortul de aprofundare ce l-ați depus pînă acum, va propunem o schiță mai mult sumara pentru a construi un editor grafic.

```
LIB.H
#define         UP      72
#define         DOWN    80
#define         RIGHT   77
#define         LEFT    75
#define         F1      59
#define         F2      60
#define         F3      61


LIB_APL.C
#include        <stdio.h>
#include        "lib.h"

main()
{
    c = getch();
    switch(c)
        {case 0 : c = getch();
                  switch(c)
                  {case UP: y -= step_y;
                            echo_update(nr,x,y);
                            break;
                  case DOWN : y += step_y;
                            echo_update(nr,x,y);
                            break;
                  case LEFT : x -= step_x;
                            echo_update(nr,x,y);
                            break;
                  case RIGHT: x += step_x;
                            echo_update(nr,x,y);
                            break;
                  case F1 : draw(x,y);
                            break;
                  case F2 : move(x,y);
                            break;
                  }
        case '1': line_color_index(3);
                  break;
        }
```

Actualitatea PC

Doru P. TIBERIU

TASTATURA PC

Tastatura calculatoarelor din familia PC XT și AT este o unitate separată fizic și logic, echipată cu un procesor propriu. Există mai multe tipuri de tastaturi, cu un număr diferit de taste (de obicei 83, 84 sau 101) amplasate diferit. Cele mai des întîlnite sînt tastaturile cu 84 de taste pentru PC AT.

Către sistem sînt expediate codurile tastelor și un cod standard al caracterelor înscrise pe ele. Modificarea codurilor în ASCII extins o realizează procedurile BIOS.

Fiecare tastă are un cod de 8 biți (scan code). Subsistemul expediază 2 coduri diferite la apăsarea sau eliberarea tastei (make/break scan code). La tastatura PC XT codul de eliberare se obține prin adăugarea a 80H la codul de apăsare. În cazul AT codul de eliberare are doi bytes — codul FOH și codul tastei apăsate. La apăsarea mai multor taste, se expediază doar codul ultimei taste apăsate. Dacă legătura cu calculatorul este blocată, în buffer-ul procesorului tastaturii se memorează doar primul cod de apăsare.

Interfața tastaturii

Tastatura este conectată cu calculatorul printr-un cablu cu 4 fire (+5 V DC, masa, CLOCK și DATA). Menținerea stării „1" pe linia CLOCK peste 200 ms produce inițializarea tastaturii; aceasta confirmă cu AAH pregătirea. Calculatorul poate întrerupe transmisia prin fixarea stării „0" pe linia CLOCK peste 60 ms (starea liniei DATA nu are importanță în acest caz). Înainte de expedierea datelor, tastatura controlează starea liniilor:

— dacă linia CLOCK este în stare „0", transmisia este blocată și datele sînt memorate în buffer;

— dacă linia CLOCK este în stare „1" și DATA în stare „0", datele sînt memorate în buffer și tastatura începe recepționarea informației;

— cînd liniile CLOCK și DATA sînt în stare „1", tastatura poate începe transmisia de date.

Schema tastaturii

Schema prezentată este un exemplu tipic de soluție constructivă. Elementul principal este procesorul 8049 Intel, cu următoarele funcții:

1) executare autotest (basic assurance test) după conectarea alimentării sau la eventuala cerere a sistemului. După test (600—900 ms) se transmite AAH (funcționare corectă) sau FCH în caz contrar;

2) comanda transmisiei seriale între tastatură și sistem prin circuitul U_1;

3) comanda protocolului reciproc de confirmare a datelor expediate (hand shake);

4) acumularea în buffere a minimum 16 coduri;

5) comanda LED care indică starea tastaturii sau a calculatorului CAPS LOCK, NUM LOCK și SCROLL LOCK (PC AT) sau POWER ON (PC XT).

Dacă construcția pare simplă, toate aceste probleme complicate le rezolvă programul conținut în memoria 8049.

Comunicarea cu sistemul PC AT

a) Comenzi de la sistem:

— RESET — cod FFH — cerere de inițializare și autotest tastatura. Aceasta confirmă cu ACK și așteaptă trecere liniilor CLOCK și DATA în stare „1";

— RESEND — cod FEH — repetarea ultimei date, eronată probabil;

— NOP — no operation — cod de la F7H la FDH;

— SET DEFAULT — cod F6H — fixarea parametrilor standard; tastatura răspunde ACK și anulează buffer-ul de ieșire, stabilește valorile inițiale și citește tastele;

— ENABLE — cod F4H — asemănătoare SET DEFAULT;

— DEFAULT DISABLE — cod F5H — după stabilirea valorilor inițiale, tastatura trece în stare pasivă;

— SET TYPE MATIC RATE/DELAY — cod F3H; răspuns ACK, se întrerupe citirea tastelor și se așteaptă un parametru (un byte):

— $b_7 = 0$

— b_6, b_5 întîrzierea: $(1 + b_6b_5) \times 250$ ms—20%; inițial continuu apasata este de (500 ms—20%)

— b_4-b_0 frecvența repetarii $(8 + b_2b_1b_0) \times 2^{b_4b_3} \times 0{,}00417$ s; inițial — 10 semne/s; poate varia intre 2 și 30 repetari/s pentru tasta continuu apasata

— ECHO — cod EEH — la cererea sistemului, se trimite EEH;

— SET/RESET MODE INDICATORS — cod FDH; se confirmă cu ACK și se așteaptă un byte care reflectă starea indicatorilor LED (V. funcția 5 a procesorului).

b) Comenzi de la tastatură:

— RESEND — cod FEH;

— confirmare — cod FAH — se expediază ACK;

— OVERUN — cod 00H — buffer-ul memoriei complet ocupat; se lansează semnal sonor;

— DIAGNOSTIC FAILURE — cod FDH sau FCH — la descoperirea greșelilor de citire a tastaturii. Dacă eroarea apare în timpul textului de bază, tastatura oprește citirea tastelor și așteaptă comenzile de la sistem sau inițializarea. În celelalte cazuri, se citește în continuare matricea tastelor;

— BREAK CODE PREFIX — cod FOH — înaintea codului tastei, la eliberarea acesteia;

— BASIC ASSURANCE TEST COMPLETION CODE — cod AAH — terminarea corectă a testului tastaturii;

— ECHO — cod EEH — răspuns la comanda ECHO a sistemului.

Procedurile de deservire a tastaturii

Datele de ieșire sînt coduri de un byte pentru semne sau coduri extinse (Extended ASCII). În tabel se prezintă codurile ASCII transmise de BIOS sistemului operațional sau programelor utilizator. În cazul funcțiilor care nu se pot reda în ASCII, se folosesc coduri extinse de 2 bytes, indicate, de asemenea, în tabel.

Anumite combinații de taste produc funcții netipice:

— ALT, CTRL, DEL simultan = system reset;

— CTRL, BREAK simultan = break;

— CTRL, NUM LOCK = pause: deblocare datorită oricărui cod, cu excepția NUM LOCK;

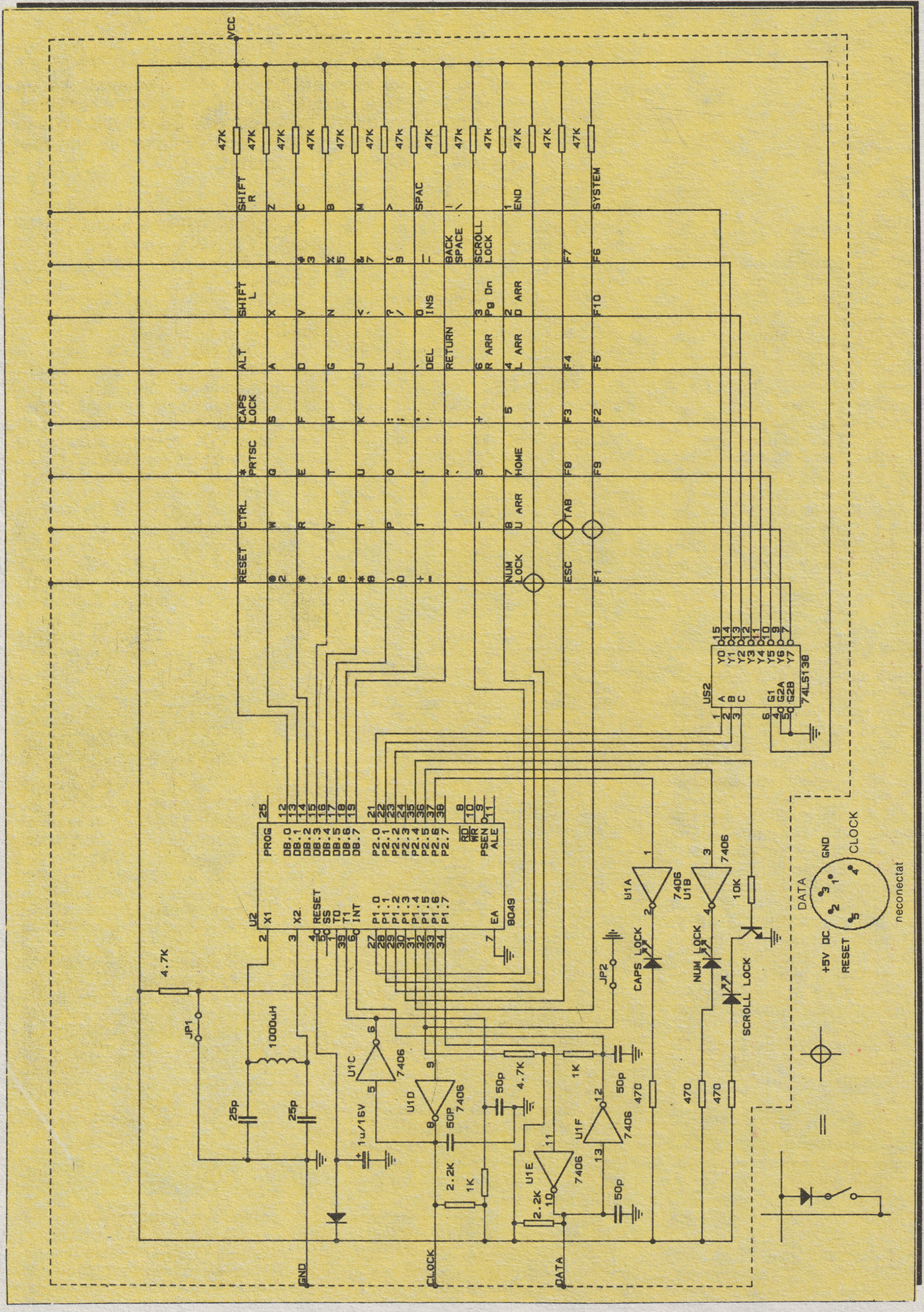
— SHIFT, PRTSC = print screen.

Exploatare

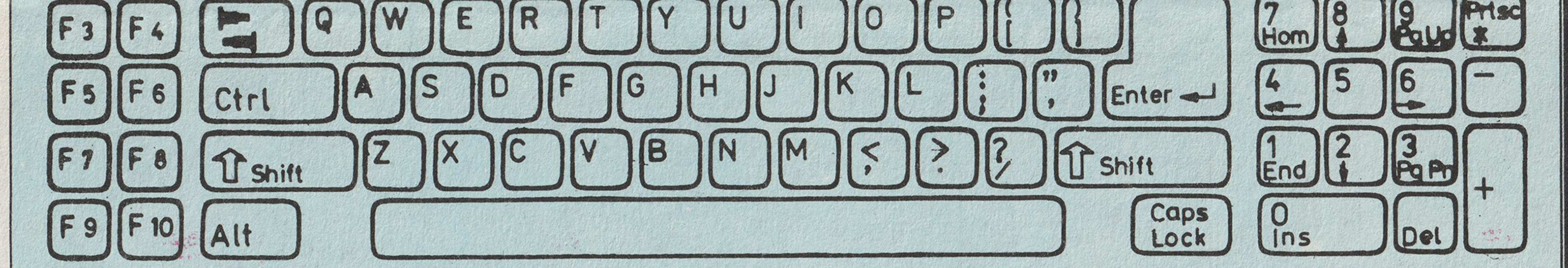
— Atenție la caracteristicile mecanice ale tastelor și carcasei.

— Atenție la durabilitatea înscrierii semnelor; cele mai bune au două straturi de material plastic diferit colorate. O bună tastă rezistă la peste 20 milioane acționări.

Spot


VCC
8049
74LS138
GND
CLOCK
DATA
+5V DC
RESET
neconectat





Tastatura cu 84 de taste

3B/BB	3C/BC	29/A9	02/82	03/83	04/84	05/85	06/86	07/87	08/88	09/89	0A/8A	0B/8B	0C/8C	0D/8D	2B/AB	0E/8E	01/81	45/C5	46/C6	••
3D/BD	3E/BE	0F/8F	10/90	11/91	12/92	13/93	14/94	15/95	16/96	17/97	18/98	19/99	1A/9A	1B/9B	•	47/C7	48/C8	49/C9	37/B7	
3F/BF	40/C0	1D/9D	1E/9E	1F/9F	20/A0	21/A1	22/A2	23/A3	24/A4	25/A5	26/A6	27/A7	28/A8	1C/9C	4B/CB	4C/CC	4D/CD	4A/CA		
41/C1	42/C2	2A/AA	2C/AC	2D/AD	2E/AE	2F/AF	30/B0	31/B1	32/B2	33/B3	34/B4	35/B5	36/B6	4F/CF	50/D0	51/D1	4E/CE			
43/C3	44/C4	38/B8	39/B9	3A/BA	52/D2	53/D3														

Codurile tastaturii PC XT

05	06	0E	16	1E	26	25	2E	36	3D	3E	46	45	4E	55	5D	66	76	77	7E	84
04	0C	0D	15	1D	24	2D	2C	35	3C	43	44	4D	54	5B	•	6C	75	7D	7C	
03	0B	14	1C	1B	23	2B	34	33	3B	42	4B	4C	52	5A	6B	73	74	7B		
83	0A	12	1A	22	21	2A	32	31	3A	41	49	4A	59	69	72	74	79			
01	09	11	29	58	70	71														

Codurile tastaturii PC AT

Spot

BIOS-interpretare coduri taste

Nr. cheie	Bază	Cu Shift	Cu Ctrl	Cu Alt
1	'	–	-1	-1
2	1	!	-1	(1)
3	2	@	Nul(000)(1)	(1)
4	3	#	-1	(1)
5	4	$	-1	(1)
6	5	%	-1	(1)
7	6	f	RS(030)	(1)
8	7	&	-1	(1)
9	8	*	-1	(1)
10	9	(	-1	(1)
11	10	)	-1	(1)
12	-	-	US(031)	(1)
13	=	+	-1	(1)
14	/	:	PS(028)	-1
15	BS(008)	(BS 008)	Del (127)	-1
16	→(009)	←-(1)	-1	-1
17	q	Q	DCI(017)	(1)
18	w	W	ETB(023)	(1)
19	e	E	ENQ(005)	(1)
20	r	R	DC2(018)	(1)
21	t	T	DC4(020)	(1)
22	y	Y	EM(025)	(1)
23	u	U	NAK(021)	(1)
24	i	I	HT(009)	(1)
25	o	O	SI(015)	(1)
26	p	P	DLE(016)	(1)
27	[	{	Esc(027)	-1
28	]	}	GS(029)	-1
30Ctrl	-1	-1	-1	-1
31	a	A	SØH(001)	(1)
32	s	S	DC3(013)	(1)
33	d	D	EOT(004)	(1)
34	f	F	ACK(006)	(1)
35	g	G	BEL(007)	(1)
36	h	H	BS(008)	(1)
37	j	J	LF(010)	(1)
38	k	K	VT(011)	(1)
39	l	L	FF(012)	(1)
40	;	:	-1	-1
41	,	,,	-1	-1
43 CR Enter	CR Enter(010)	LF	-1	
44 Shift stînga	-1	-1	-1	-1
46	z	Z	SUB(026)	(1)
47	x	X	CAN(024)	(1)
48	c	C	EXT(003)	(1)
49	v	V	SYN(022)	(1)
50	b	B	STX(002)	(1)
51	n	N	SO(014)	(1)
52	m	M	CR(013)	(1)
53	,	‘	-1	-1
54	.	>	-1	-1
55	/	?	-1	-1
57 Shift dreapta	-1	-1	-1	-1
58 Alt	-1	-1	-1	-1
61	SP	SP	SP	SP
64 Caps Lock	-1	-1	-1	-1
65	Nul(1)	Nul(1)	Nul(1)	Nul(1)
66	Nul(1)	Nul(1)	Nul(1)	Nul(1)
67	Nul(1)	Nul(1)	Nul(1)	Nul(1)
68	Nul(1)	Nul(1)	Nul(1)	Nul(1)
69	Nul(1)	Nul(1)	Nul(1)	Nul(1)
70	Nul(1)	Nul(1)	Nul(1)	Nul(1)
71	Nul(1)	Nul(1)	Nul(1)	Nul(1)
72	Nul(1)	Nul(1)	Nul(1)	Nul(1)
73	Nul(1)	Nul(1)	Nul(1)	Nul(1)
74	Nul(1)	Nul(1)	Nul(1)	Nul(1)
90	Esc	Esc	Esc	-1
95 Num Lock	-1	-1	Pause(2)	-1
100 Scroll	-1	-1	Break (2)	-1

Nr. cheie	cu Num Lock	Sau de bază	cu ALT	cu CTRL
91	7	Home(1)	-1	clear ecran
92	4	←-(1)	-1	înapoi un cuvînt(1)
93	1	End(1)	-1	clear o linie(1)
96	8	Up(1)	-1	-1
97	5	-1	-1	-1
98	2	Down(1)	-1	-1
99	0	Ins	-1	-1
101	9	Page Up (1)	-1	la început de text
102	6	→(1)	-1	cuvînt înainte(1)
103	3	Page Down (1)	-1	clear ecr(1)
104	.	Del(1)(2)	(2)	(2)
106	*	*	-1	Nul(1)
107	–	–	-1	-1
108	+	+	-1	-1
105	Sys Re	fără		

(1)cod extins (2)funcţii spec.

Coduri extinse

Cod Suplim. Nul	Funcţia
15	–
16-25	Alt Q,W,E,R,T,Y,U,I,O,P
30-38	Alt A,S,D,F,G,H,J,K,L
44-50	Alt Z,C,V,B,N,M
59-68	F1 la F10 taste fcţ mod bază
71	Home (colţ stînga sus)
72	Up
73	Page Up& Home
75	--
77	--
79	End
80	Down
81	Page Down& Home
82	Ins(scriere supl.)
83	Del (anulare)
84-93	F11 la F20(Shift F1 la F10)
94-103	F21 la F30(Ctrl F1 la F10)
104-113	F31 la F40(Alt F1 la F10)
114	Ctrl PrtSc(Start/stop imprim.)
115	Ctrl --(retur un cuvînt)
116	Ctrl --(avans un cuvînt)
117	Ctrl End(anulare linie)
118	Ctrl PgDn(anulare ecran)
119	Ctrl Home(clear ecran)
120-131	Alt 1,2,3,4,5,6,7,8,9,0,-,= (taste 2-13)
132	Ctrl PgUp(la început text)

Spot

Valeriu IORGA

TURBO-PASCAL versiunile 5.0 şi 5.5 (II)

1. Compilatorul TURBO-Pascal în linie de comandă

Compilatorul Pascal în linie de comandă (TPC.EXE) este identic celui folosit în mediul integrat de dezvoltare (TURBO. EXE). Lansarea compilării se face prin comanda:

TPC [opţiuni] fişier-sursă [opţiuni]

Dacă nu se indică o extensie pentru numele fişierul sursă, atunci va fi căutat un fişier cu acest nume şi extensia PAS.

Dacă fişierul sursă reprezintă un program, se va crea pe disc un fişier executabil cu acelaşi nume şi extensia EXE, iar dacă este o unitate, se va crea pe disc un fişier cu acelaşi nume şi extensia TPU.

Opţiunile se specifică prin / urmat de o literă.

Lista opţiunilor în limita de comandă

În textul programului pot fi inserate *directive de compilare* avînd una din formele:

{$directiva+} opţiune activă
{$directiva—} opţiune inactivă
{$directivă info}

Lista opţiunilor în linia de comandă

/$A	/$A—	aliniere date	cuvînt octet
/$B	/$B—	evaluare booleană	completă-optimizată
/$D	/$D—	informaţie de depanare	on-off
/$E	/$E—	emulare	on-off
/$F	/$F—	forţare apeluri lungi	on-off
/$I	/$I—	verificare I/E	on-off
/$L	/$L—	simboluri locale	on-off
/$Msss,min, max		dimensiune memorie	
/$N	/$N—	procesare numerică	coprocesor-software
/$O	/$O—	segmentare permisă	on-off
/$R	/$R—	verificare domeniu	on-off
/$S	/$S—	verificare stivă	on-off
/$V	/$V—	verificare variabile şir	on-off
/Ddefinire		definire condiţională	
/GS		fişier MAP	segmente
/GP		fişier MAP	simboluri publice
/GD		fişier MAP	detalii
/L		buffer editor legături	disc
/Ecale		director EXE & TPU	
/Icale		directoare INCLUDE,	
/Ocale		directoare OBJ	
/Tcale		director Turbo	
/Ucale		directoare unităţi	
/V		depanare separată	on
/B		opţiunea build	
/M		opţiunea make	

Directivele de compilare sînt aceleaşi ca şi opţiunile de compilare din linia de comandă:

{$B+} expresiile booleene sînt evaluate complet;
{$D+} se generează informaţie pentru depanator;
{$E+} se încorporează emulatorul;
{$F+} se generează apeluri lungi;
{$I+} erorile de I/E sînt tratate de Turbo Pascal;
{$I fişier} se inserează la compilare fişierul sursă menţionat;
{$L+} simbolurile locale sînt înregistrate în fişierul EXE;
{$L fişier} fişierul OBJ generat de asamblor este inclus la editarea de legături
{$M stiva, hmin hmax} fixează mărimea stivei, ca şi mărimea minimă şi maximă a heapului;
{$R+} controlează indicii tablourilor şi verifică compatibilitatea atribuirilor;
{$S+} efectuează testul de stivă plină, generînd în caz afirmativ o eroare de execuţie;
{$V+} verifică tipul variabilelor transmise ca parametri.

2. Programarea orientată pe obiecte în TURBO-Pascal versiunea 5.5

Versiunea 5.5 posedă următoarele caracteristici proprii limbajelor orientate pe obiecte:

— *încapsularea* — definită ca o combinaţie între o structură de date şi una sau mai multe proceduri care prelucrează aceste date, introducîndu-se astfel un nou tip de date — *obiectul.*

— *moştenirea* — un obiect B, descendent al unui obiect A, poate folosi definirile de date şi procedurile proprii obiectului A.

— *polimorfismul* — aceeaşi procedură (metoda) poate acţiona asupra unor obiecte diferite, fiecare obiect definind modul specific de aplicare a procedurii.

TURBO-Pascal 5.5 adaugă noi cuvinte rezervate în definirea limbajului, şi anume:

constructor destructor object virtual

Să considerăm, de exemplu, două tipuri înregistrare cu structuri asemănătoare:

```
type
  Poziţie = record
    X,Y: integer;
    end;
  Punct=record
    X,Y: integer;
    vizibil: boolean;
    end;
```

Un tip obiect este o structură formată dintr-un număr fixat de componente (ca şi înregistrarea). În cazul nostru:

```
type
    Poziţie=object
    X,Y: integer;
    end;
```

Un tip obiect poate moşteni componente ale altui tip obiect, acesta din urmă fiind *descendent* din primul tip (tipul *părinte* sau *strămoş).*

```
type
  Punct=object (Poziţie)
    vizibil: boolean:
    end;
```

(Continuare în pag. 29)

GHIDUL UTILIZATORULUI



TURBO - PASCAL

(Urmare din pag. 27)

Domeniul unui tip obiect este reprezentat de acel tip și de toate tipurile descendente.

Fiecare componentă a tipului obiect poate fi *un cîmp* de date sau o *metodă*, care realizează o operație asupra obiectului. Definirea obiectului presupune declararea metodei prin antetul ei. În exemplul nostru vom considera o metodă Init de inițializare a poziției:

```
type
  Poziție=object
    X,Y: integer;
    procedure Init (XO, YO: integer);
    end;
```

Declararea propriu-zisă a procedurii (metodei) se face în afara obiectului, fiind calificată cu numele tipului obiect:

```
procedure Poziție. Init (XO, YO: integer);
begin
  X:=XO;
  Y:=YO;
end;
```

Un tip obiect se declară numai într-o definire de tip, în blocul cel mai exterior al programului sau unității (nu se pot declara tipuri obiecte în secțiunea **var** sau într-un bloc al unei proceduri sau funcții).

Un obiect este *instanțiat* sau creat printr-o declarație de variabila de tip obiect:

```
var: P: Poziție;
```

sau prin aplicarea procedurii New unui pointer la un obiect:

```
var PP: ^Poziție;
...
New (PP);
```

Pentru inițializarea instantei P (sau PP^) a obiectului Poziție se apelează metoda, calificată cu numele instanței:

```
P.Init (20, 30); {sau PP^. Init (20, 30)}
```

Deoarece atît cîmpurile, cît și metodele aparțin obiectului, acesta se poate referi direct la cîmpuri, fără a utiliza instrucțiunea **with.** Putem considera că există o instrucțiune **with** implicită în metoda cu un parametru invizibil *self* (un pointer la o instanță a obiectului):

```
procedure Poziție. Init (XO, YO: integer; var self: poziție);
begin
  self.X:=XO;
  self.Y:=YO;
end;
```

Parametrii formali ai metodei nu pot fi identici cu cîmpurile de date ale obiectului. La definirea unui tip descendent pot fi folosite metodele tipului parinte, dar acestea pot fi redefinite (definind metode cu același nume). Cîmpurile de date din tipul părinte nu pot fi redefinite într-un tip descendent.

De exemplu, pentru obiectul Cerc, descendent din tipul obiect Punct, întrucît tipul obiect Cerc are un nou cîmp: Raza, inițializarea necesită o nouă metodă Init, care modifică atît raza, cît și cîmpurile moștenite (folosind metoda Init moștenită pentru Punct)

```
type
  Cerc=object(Punct)
    Raza: integer;
    procedure Init (XO, YO, RO: integer);
    end;
procedure Cerc. Init (XO, YO, RO: integer);
begin
  Punct.Init (XO, YO);
  Raza :=RO;
end;
```

Într-o ierarhie de tipuri obiecte (de exemplu Poziție, Punct, Cerc, Arc,...) obținută prin moștenire, prezintă interes metodele aplicabile tuturor tipurilor din ierarhie (de exemplu o metodă de reprezentare grafică a obiectelor).

Un tip descendent este compatibil cu toate tipurile strămoși. Această compatibilitate extinsă are trei aspecte:

— între instanțele obiectelor;
— între pointerii la instanțele obiectelor;
— între parametrii formali și actuali;

Compatibilitatea este direcționată de la descendent la strămoș și nu invers.

Astfel pentru instanțele obiectelor:

```
var
  P: Punct;
  C: Cerc;
  PP: ^Punct;
  PC: ^Cerc;
```

sînt permise atribuirile:

```
P:=C;
PP:=PC;
```

dar nu și atribuirile:

```
C:=P;
PC:=PP;
```

Un parametru formal variabil de tip obiect poate fi înlocuit cu un parametru actual de tip obiect descendent. Astfel tipul exact al parametrului actual poate fi necunoscut la compilare, el fiind unul din tipurile obiect descendente, reprezentînd așadar un *obiect polimorfic.* Polimorfismul se implementează în TURBO-Pascal 5.5 prin intermediul *metodelor virtuale.* În mod implicit metodele obiectelor sînt *statice.* Apelul unei metode statice este rezolvat la compilare, în timp ce al unei metode virtuale este amînat pînă în momentul execuției. O metodă declarată virtuală într-un tip obiect va rămîne virtuală și în tipurile descendente. Fiecare tip obiect care posedă o metodă virtuală trebuie să aibă un *constructor*, adică o metodă (statică) de tip special servind la instalarea metodelor virtuale pe care le precede.

```
type
  Punct=object (Poziție)
    vizibil: boolean;
    constructor Init (XO, YO. integer);
    procedure Desenare; virtual;
  end;
  Cerc=object (Punct)
    Raza: integer;
    constructor Init (XO, YO, RO: integer);
    procedure Desenare; virtual;
  end;
```

Fiecare instanță a unui obiect ce folosește metode virtuale trebuie inițializată prin apelarea unui constructor. Antetele pentru metodele virtuale cu același nume din tipurile descendente trebuie să fie identice cu același număr de parametri.

Revenind la exemplul cu figurile geometrice, acestea au în comun atributele obiectului Punct, adică:

— o poziție (X, Y) numită *punct de reper*
— un mod vizibil/invizibil

Atributele comune vor fi colectate într-un singur tip, ierarhia de tipuri obiecte moștenind aceste proprietăți. Un obiect care are caracteristici ce vor fi moștenite de toate obiectele descendente reprezintă un tip *obiect abstract.* Se pot adăuga noi metode tipului obiect abstract, care vor fi implementate în mod diferit în tipurile descendente folosind metodele virtuale.

Pentru obiectele dinamice (alocate în heap și manipulate prin pointeri) care conțin metode virtuale au fost introduse facilități noi de alocare/dealocare. Astfel procedura New asigură alocarea de spațiu pentru un obiect dinamic în heap și inițializarea obiectului prin apelarea unui constructor. Astfel:

```
New (PC, Init (60, 50, 30));
```

este echivalent cu:

```
New (PC);
PC^.Init (60, 50, 30);
```

New a fost de asemenea extinsă, permițînd apelul ca o funcție, care întoarce o valoare de tip pointer.

Un obiect dinamic care nu mai este necesar va fi dealocat printr-o metodă mai sofisticată decît procedura Dispose, numită metoda *destructor.* Destructorul este definit alături de celelalte metode în tipul obiect. Întrucît sînt necesare diferite acțiuni pentru dealocarea diferitelor obiecte, se recomandă definirea destructorilor ca metode virtuale, fără parametri. Astfel pentru obiectul:

```
type
  Punct=object(Poziție)
    vizibil: boolean;
    constructor Init (XO, YO: integer);
    destructor Done; virtual;
    procedure Desenare; virtual;
  end;
```

pentru instanțierea:

```
var PP: ^Punct;
```

dealocarea obiectului se poate face cu:

```
Dispose (PP,Done);
```

echivalentă cu:

```
PP^.Done;
Dispose (PP);
```

Programarea orientată pe obiecte este un răspuns direct la complexitatea aplicațiilor moderne. Încapsularea și moștenirea sînt instrumente eficace pentru stăpînirea acestei complexități.

În mediul de dezvoltare integrată TURBO-Pascal, depanatorul TURBO-Debugger a fost îmbunătățit în versiunea 1.5, pentru a permite depanarea programelor orientate pe obiecte prin examinarea obiectelor și trasarea execuției metodelor.

Stelian NICULESCU,
Cristian ARTEMI

MS—DOS (II)

În cele ce urmează vor fi prezentate succint toate comenzile sistemului de operare MS—DOS 4.0. Schema de prezentare cuprinde:

● Numele comenzii, care în general este o prescurtare a cuvîntului din limba engleză ce descrie acțiunea comenzii.

● I dacă comanda este internă (se află în fișierul Command. Com).

E dacă comanda este internă procesorului de comenzi (se află într-un fișier separat).

● Scop: precizează la ce folosește comanda.

● Sintaxă: numecomandă [opțiuni].

Opțiunile pot fi: disc, cale, nume-cale, nume-fișier, comutator sau argument. Calea este o succesiune de directoare despărțite de caracterul ? ce desemneaza un drum într-un arbore director. Nume-cale este o cale la care se adauga numele unui fișier. Opțiunea disc specifica o unitate de disc. Comutatoarele controlează acțiunea comenzii. Ele încep cu simbolul / (exemplu /p). Argumentele furnizează informații necesare comenzii. De obicei trebuie ales între două sau mai multe argumente, de exemplu, între on sau off.

Parantezele drepte din descrierea sintactică a unei comenzi semnifică opționalitatea conținutului, iar succesiunea de trei puncte semnifică repetitivitatea.

● Comentarii: explică comanda și opțiunile din punct de vedere funcțional.

● Exemple: Exemplifică utilizarea comenzii.

În ordine alfabetică comenzile sînt:

1. **Append E**

● Scop: Fixează o cale de căutare pentru fișierele de date.

● Sintaxa la prima utilizare:

append [/x] [/e]

Pentru a specifica directoarele căutate:

append [disc:] cale [; [disc:] [cale] ...]

Pentru a șterge căile fixate:

append;

unde cale este directorul în care MS-DOS caută fișierul de date.

● Comentarii: Comanda append permite specificarea unei căi pentru fișiere de date. Comanda append acceptă indicatori doar cînd este folosită prima dată.

/x Extinde căutarea pentru fișiere de date. MS-DOS caută mai întîi fișierele de date în directorul curent. Pe cele pe care nu le găsește aici le caută în directoarele anexate prin comanda append

/e Face ca directoarele anexate să fie memorate în mediul MS-DOS.

Pot fi specificate mai multe căi separate prin (;).

Dacă se tastează comanda append, cu opțiunea de cale, a doua oară, MS-DOS anulează vechea cale de căutare și o folosește pe cea nouă.

Dacă nu se folosesc opțiuni MS-DOS afișează calea de date curentă. Dacă se folosește comanda append; MS-DOS anulează căile de date anexate anterior. altfel zis caută fișierele de date doar în directorul de lucru.

● Exemple:

Pentru a căuta fișierele de date în directorul scrisori de pe discul C și în directorul rapoarte de pe discul A folosim comanda:

append C:\scrisori; a:\rapoarte

Dacă dorim ca MS-DOS să caute întîi în directorul curent și apoi în căile de date anexate, atunci înaintea comenzii prezentate se va utiliza:

append /x

2. **Assign E**

● Scop: Asociază o literă de unitate de disc la o altă unitate de disc.

● Sintaxă:

assign [x[=] y[...]]

unde x este unitatea din/în care MS-DOS citește/scrie, iar y este unitatea din/în care vrem ca MS-DOS să citească/scrie.

● Comentarii: Comanda assign ne permite să citim și să scriem fișiere pe alte discuri decît A și B, pentru aplicații care utilizează doar aceste două unități. Nu poate fi asociat un disc nedefinit (inexistent). Deoarece assign ascunde adevăratul tip al unității, nu trebuie folosit:

— cu comenzi care cer informații despre disc (backup, restore, label, join, subst, print);

— în timpul utilizării normale a MS-DOS-ului, exceptînd situația cînd utilizarea este cerută de un program. Alte două comenzi, format și diskcopy, ignoră reasocierile unităților de disc.

● Exemple:

Pentru a reface asocierile originale se folosește comanda:

assign

Dacă dorim să executăm o aplicație pe discul rigid C și această aplicație necesita să punem programul pe disc în unitatea A și discul de date în unitatea B, se folosește comanda:

assign a=c b=c

Toate referirile la discurile A și B vor merge la discul C.

3. **Attrib E**

● Scop: Pune și afișează atributele de fișier.

● Sintaxă:

attrib [± r] [± a] [disc:] nume-cale [/s]

unde:

+ r pune atributul doar pentru citirea unui fișier;

- r dezafectează modul doar-citire;

+ a pune atributul de arhivă unui fișier;

- a șterge atributul de arhiva;

● Comentarii: Comanda attrib pune atributul doar-citire și/sau atributul de arhiva pentru fișiere. Discul și calea specifică localizarea fișierului sau fișierelor la care se face referință. /s extinde acțiunea la toate subdirectoarele. Comenzile backup, restore și xcopy folosesc atributul de arhiva ca un mecanism de control. Se pot folosi opțiunile + a și -a daca dorim să folosim aceste comenzi.

● Exemple:

Pentru a afișa atributul unui fișier news.doc de pe discul C se folosește comanda:

attrib C: news.doc

Următoarea comanda atribuie fișierului report.txt permisiunea doar de citire:

attrib + r report.txt

Punînd unui fișier atributul doar-citire, se previn ștergerile și modificarile accidentale. Pentru a șterge atributul doar-citire pentru fișierele din directorul\user\pete de pe discul A și pentru fișierele din subdirectoare, trebuie folosită comanda:

attrib - r a:\user\pete /s

Dacă se dorește copierea pe un disc B a tuturor fișierelor de pe un disc A, cu excepția celor de extensie .bak, se folosesc comenzile:

attrib + a a : * *

attrib - a a : *.bak

xcopy a: b:/m

4. **Backup E**

● Scop: Pune unul sau mai multe fișiere de pe un disc pe altul.

Pentru a le utiliza, fișierele trebuie restaurate.

● Sintaxă:

backup [disc 1:] [cale] [numefișier] [disc 2:] [/s] [/m] [/a] [/f] [/d:data] [/t:timp] [/L: [disc:] [cale] [numefișier]]

unde disc 1 este unitatea de disc care va fi reprodusa și disc 2 este unitatea de destinație.

● Comentarii: Backup poate lucra cu discuri de tip diferit (flexibile sau rigide) și de densitate diferită. Indicatorii au următoarele semnificații:

/s reproduce și subdirectoarele

/m reproduce doar fișierele care au fost modificate de la ultima comandă backup

/a adaugă fișierele pe discul destinație la cele existente fără a le distruge pe cele din urmă

/f duce la inițializarea discului destinație daca acesta nu este deja inițializat. Pentru aceasta fișierul format.com din MS-DOS trebuie să fie accesibil din calea curenta

/d: data reproduce doar fișierele care au fost modificate la data sau ulterior

/t: timp reproduce doar fișierele care au fost modificate la sau după timp
/L: numefișier face o intrare backup.log în fișierul specificat. Dacă nu se specifica numefișier, backup pune un fișier numit backup.log în directorul rădăcină al discului care conține fișierele ce sînt reproduse.
Un fișier backup.log folosește formatul:
— prima linie conține data și timpul la care s-a făcut operația de transfer;
— cîte o linie pentru fiecare fișier care conține numele fișierului și numărul discului pe care se află.
Dacă fișierul backup.log există deja, comanda adaugă la el intrarea curentă.
• Exemplu:
Pentru a transfera toate fișierele din directorul \user\emily, de pe discul C, pe un disc gol și formatat A, folosim comanda: backup C:\user\emily a:

5. **Break** I
• Scop: Activează verificarea secvenței CONTROL-C.
• Sintaxa:
break [on]
sau break [off]
• Comentarii: Funcție de programul care rulează se poate utiliza CONTROL-C pentru oprirea unei activități (de exemplu sortarea unui fișier).
În mod normal MS-DOS verifică dacă a fost apăsată secvența CONTROL-C în timp ce citește de la tastatură sau scrie pe ecran sau la imprimantă. Dacă se activează break on, se extinde funcționarea lui CONTROL-C și pentru alte funcții, de exemplu: citire/scriere de pe/pe disc. Unele programe își activează singure modul de răspuns la CONTROL-C. Punerea lui break pe on nu afectează aceste programe.
Folosirea sintaxei break off are ca efect anularea acțiunii lui break on. Pentru a afla cum este poziționat break (pe on sau pe off) se folosește sintaxa break.

6. **Chcp** I
• Scop: Afișează sau schimbă codul curent de pagină pentru procesorul de comenzi command.com. (Codul de pagină este un tabel care definește mulțimea caracterelor folosite și este specific țării și limbajului utilizat.)
• Sintaxa:
chcp [nnn]
unde nnn este codul de pagină.
• Comentarii: Comanda chcp acceptă unul din cele două coduri de pagini sistem pregătite. Se afișează un mesaj de eroare dacă se selectează un cod de pagină care nu a fost pregătit pentru sistem.
Dacă se dă comanda chcp fără codul de pagină, se afișează pe ecran codul de pagină activ și codurile de pagină pregătite pentru sistem.
Codurile de pagină valide sînt:
437 Statele Unite
850 Multinational
860 Portugheză
863 Franceză—Canadiană
865 Nordic
Programele rulate după schimbarea codului de pagină vor folosi noul cod. Pentru a vedea codul de pagină curent se folosește sintaxa chcp.

7. **Chdir (cd)** I
• Scopul: Schimbă un director cu o altă cale sau afișează directorul de lucru.
• Sintaxa:
chdir [cale] sau cd [cale]
• Comentarii: Comanda chdir schimbă directorul de lucru cu directorul specificat prin cale. Pentru a se afișa numele directorului de lucru se folosește sintaxa cd sau chdir.
• Exemple:
Comenzile chdir\nivel și cd nivel au aceeași acțiune: schimbă directorul curent în directorul nivel. Se pot folosi și prescurtări. Dacă directorul curent este directorul\nivel, pentru a se ajunge în directorul\nivel\nivel 1 putem folosi comanda:
cd nivel 1 în loc de
cd\nivel\nivel 1
Apoi dacă dorim să ne întoarcem în directorul părinte putem folosi simbolurile .. ce reprezintă numele părintelui directorului curent.
cd..

8. **Chkdsk** E
• Scop: Parcurge discul din unitatea de disc specificată și îl verifică de erori.
• Sintaxa:
chkdsk [disc:] [numecale] [/f] [/v]
• Comentarii: Comanda chkdsk arată starea discului specificat. Dacă chkdsk găsește erori pe disc, afișează mesajele de eroare urmate de starea discului. Dacă la comanda chkdsk se dă prin numecale numele unui fișier, MS-DOS afișează un raport pentru disc și fișierul specificat.
Semnificația comutatoarelor este:
/f Repară erorile de pe disc. Dacă nu se specifică acest comutator, chkdsk nu corectează erorile găsite.
/v Afișează numele fiecărui fișier din fiecare director, în timpul testării discului.
Dacă chkdsk raportează un mare număr de sectoare defecte, discul trebuie reparat (reinițializat).
• Observație: chkdsk nu funcționează pe unități de disc utilizate în comenzile subst sau join.

9. **Cls** I
• Scop: Curăță ecranul
• Sintaxă:
cls
• Comentarii: Comanda cls curăță ecranul, lăsînd numai prompter-ul MS-DOS și cursorul.
• Exemplu:
Dacă se dorește începerea unui nou proces cu un ecran curat se tastează cls

10. **Command** E
• Scop: Pornește un nou proces de comenzi.
• Sintaxa:
command [disc:] [cale] [ctty-dev] [/e:nnnnn] [/p] [/c șir]
unde ctty-dev permite specificarea unui dispozitiv diferit pentru intrare sau ieșire.
• Comentarii: Procesorul de comenzi este încărcat în memorie în două părți: tranzitorie și rezidentă. În timpul rulării unele aplicații pot să modifice partea tranzitorie a procesorului. Cînd se întîmplă acest fenomen, partea rezidentă a procesorului de comenzi caută fișierul command.com pe disc pentru a reîncărca partea tranzitorie.
Opțiunea [disc:] [cale] arată procesorului de comenzi unde să caute fișierul command.com. Comutatoarele au următoarele semnificații:
/e:nnnnn Specifică mărimea spațiului de memorie a mediului MS-DOS, unde nnnnn este mărimea în octeți, de la 160 la 32768. Valoarea implicită este 160.
/p Păstrează al doilea procesor de comenzi în memorie și nu se reîntoarce automat spre primul procesor de comenzi.
/c șir Cere procesorului de comenzi să execute comanda sau comenzile specificate de șir și apoi să se întoarcă automat la primul procesor de comenzi.
• Exemplu:
Comanda de mai jos efectuează următoarele acțiuni:
— pornește un nou procesor de comenzi sub programul curent;
— rulează comanda chkdsk b:
— returnează controlul spre primul procesor de comenzi;
command /c chkdsk b:

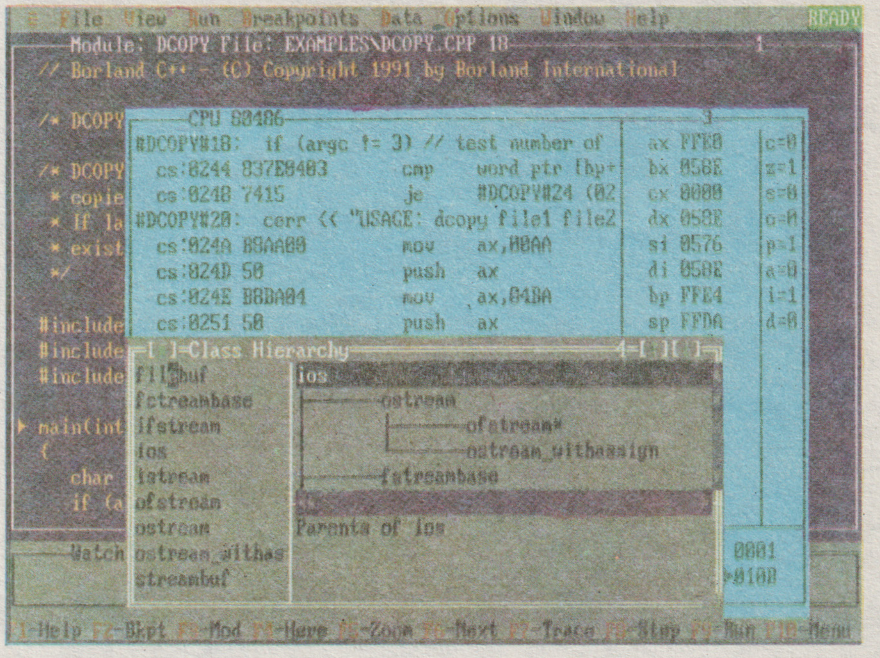
11. **Comp** E
• Scop: Compară conținutul a două seturi de fișiere.
• Sintaxa:
comp [disc 1:] [cale 1] [disc 2:] [cale 2]
• Comentarii: Comanda comp compară un fișier sau un set de fișiere (calea 1) cu un al doilea fișier sau set de fișiere (calea 2).
Dacă se precizează doar disc 2 fără cale 2, atunci cale 2 este considerată egală cu cale 1. În timp ce rulează, comp afișează calea și numele fișierelor comparate. Dacă comanda nu găsește un fișier conform cu calea primită, apare un mesaj de eroare. În timpul comparării apare cîte un mesaj pentru fiecare pereche de locații din cele două fișiere care conțin informații diferite. Mesajul indică zona în care s-a produs neconcordanța, precum și conținutul octeților diferiți. După 10 neconcordanțe, compararea se oprește.
Dacă fișierele sînt de mărime diferită, este întrebat utilizatorul dacă dorește continuarea comparării. În cazul unui răspuns pozitiv, se compară fișierele pînă la sfîrșitul celui mai scurt.
Dacă compararea este reușită, comp afișează următorul mesaj: Files — compare OK. Cînd se încheie compararea ultimei perechi de fișiere specificate se afișează mesajul: Compare more files (Y/N)? Dacă se tastează Y se reia execuția comenzii comp cerînd opțiuni noi de căi. Dacă se tastează N se încheie execuția.

12. **Copy** I
• Scop: Copiază unul sau mai multe fișiere într-o altă locație. Această comandă are și rolul de a concatena (lipi) fișiere.
• Sintaxa:
Pentru copierea fișierelor:
copy [disc 1:] numecale 1 [disc 2:] [numecale 2] [/v] [/a] [/b] sau
copy [disc 1:] numecale 1 [/v] [/a] [/b] [disc 2:] [numecale 2]
Pentru lipirea fișierelor:
copy numecale 1 + numecale 2[...] numecale n
• Comentarii: Dacă utilizatorul nu specifică numecale 2, copia este creată în directorul de lucru de pe discul curent. Această copie are același nume, aceeași dată și timp ca fișierul sursă (numecale 1). Dacă fișierul sursă este pe discul curent și utilizatorul nu specifică numecale 2, comanda copy nu se execută (nu poate copia un fișier peste el însuși) și MS-DOS afișează un mesaj de eroare.
Comanda de copiere acceptă următoarele comutatoare:
/v Determină verificarea corectitudinii sectoarelor scrise pe discul destinație.
/a Permite copierea fișierelor codificate ASCII.
/b Permite copierea fișierelor binare.
Comanda copy permite concatenarea fișierelor. Pentru aceasta se numesc fișierele (separate cu semnul +) ca opțiuni și apoi se specifică un fișier destinație spre care se trimite fișierul combinat.

(Continuare în pag. 33)

BORLAND C++ 2.0

Peter SHULTZ (INFO WORLD, 18 martie 1991)



Ultima versiune de C++ de la Borland vă oferă tot ceea ce este necesar pentru a dezvolta aplicații sub Windows 3.0. Noi considerăm aceasta ca un extraordinar mediu de dezvoltare și plăcerea e dublă deoarece el lucrează complet independent de Microsoft Windows Software Development kit (SDK).

Prin această realizare Borland a creat două linii de produse atît pentru nivelul cel mai înalt, cît și pentru utilizatorii de nivel mai scăzut. Produsul C++ 2.0 care va fi prezentat în continuare este produsul de vîrf care înlocuiește produsul Turbo C++ Professional. El este vîndut la prețul de 495.95 $. Nivelul scăzut este reprezentat prin noul produs, denumit Turbo C++ Second Edition, care este o nouă versiune a lui Turbo C++ 1.01 și care are un preț de desfacere cu amănuntul de 99.95 $.

FACILITĂȚI:

Un sistem complet de dezvoltare a aplicațiilor utilizator Borland C++ 2.0 cuprinde două compilatoare: unul este în concordanță cu produsul C++ 2.0 de la AT & T, iar celălalt este 100% compatibil cu standardul ANSI-C. Ambele au numeroase extensii de limbaje.

Dar acea caracteristică remarcabilă a acestei versiuni este suportul pentru Windows 3.0. Deși mediul de dezvoltare nu este el însuși o aplicație Windows (el nu poate rula sub Windows decît într-o fereastră DOS), el conține tot ceea ce este necesar pentru a construi programe capabile să ruleze în acest mediu. De altfel, Borland chiar a luat licența de la Microsoft pentru Resource Compiler și pentru **header**-ele Windows pe care le-a inclus în pachet. **(Microsoft Help Compiler)** nu a fost inclus în pachetul inițial, dar Borland a luat deja licența și îl va trimite gratis acelor utilizatori care nu îl au.)

De asemenea, Borland mai include o altă piatră prețioasă în cutie: pachetul **Whitewater Resource Toolkit.** Acesta vă oferă mijloacele pentru a construi și modifica interactiv resursele pentru aplicațiile Windows. Windows Resources pot fi memorate în formă binară în mai multe versiuni: executabilă (EXE), **Dynamic Link Libraries** (DLL), resurse (RES), **bit map** (BMP), cursor (CUR) sau editare/compilare/linkare care durează enorm cînd vă aflați în faza de dezvoltare a resurselor. Resursele pot fi, de asemenea, salvate în mod text care este compatibil cu fișierele **Windows Resource Script** (RC).

Cu ajutorul pachetului **Whitewater Resource Toolkit** puteți să creați și să modificați orice resursă fără să părăsiți mediul Windows. Spre deosebire de **Microsoft SDK,** trebuie să editați fișierele de text sub DOS, să creați fișierele RC referindu-le pe acelea și apoi să rulați compilatorul pentru resurse. Apoi reintrați sub Windows și testați resursele, repetînd ciclul pînă ce ajungeți la rezultatul dorit. Suplimentar la aceasta **Resource Toolkit** vă permite să testați interactiv **menu**-urile.

Pentru a putea dezvolta programe mari, Borland C++ 2.0 aduce ca noutate versiunea de lucru în mod protejat pentru Programmer's Platform, compilatorul utilizabil direct din linia de comandă DOS și, de asemenea, produsul Turbo Assembler. Acestea pot lucra sub Windows într-o fereastră DOS (chiar și în modul standard) putîndu-se dezvolta aplicații mai mari și utilizînd intensiv simbolurile de o manieră ce nu era posibilă la versiunile anterioare.

Această versiune aduce, de asemenea, suportul pentru fișierele **header** deja precompilate și un asamblor încorporat. În mod normal, cînd compilați o aplicație, o bună parte din timp se pierde compilînd fișierele **header.** Acest lucru este cu atît mai deranjant în cazul aplicațiilor Windows cînd fișierul de bază WINDOWS.H are o dimensiune de 120 kB. Deoarece cele mai multe header-e nu suferă modificări între compilări, Borland a introdus la C++ 2.0 capacitatea de a reutiliza tabelele de simboluri deja compilate din aceste **header-e.**

Versiunile anterioare de C++ de la Borland necesitau folosirea unui asamblor extern atunci cînd se făcea inserția de limbaj de asamblare în codul sursă C. Versiunea C++ 2.0 conține propriul asamblor.

Programmer's Platform este un nou mediu integrat de dezvoltare (IDE-Integrated Development Environment) creat de Borland. Cu ajutorul lui se pot acum crea aplicații Windows la fel de ușor ca și aplicațiile DOS obișnuite. Toate utilitarele de dezvoltare pentru programe Windows sînt, de asemenea, integrate, cu excepția pachetului Resource Toolkit, oferind un mediu de dezvoltare fără diferențe între Windows și DOS. Resource Toolkit va fi utilizat ca o aplicație Windows separată.

Programmer's Platform a fost, de asemenea, îmbogățit cu facilități de anulare și reluare a unei operații, facilități cu mare siguranță în funcționare. Versiunea anterioară de IDE permitea un singur nivel de anulare, pe cînd acum se poate anula sau relua selectiv aproape orice acțiune de editare ce a avut loc.

Turbo Debugger-ul a fost îmbogățit pentru lucrul cu Windows. Spre deosebire de alte debugger-e Windows, acesta poate lucra cu un singur monitor. De asemenea pot fi utilizate și regimuri de lucru cu două monitoare sau lucrul de la un alt terminal cuplat pe linie serială. **Turbo Debugger**-ul poate depana aplicații Windows, biblioteci DLL și, de asemenea, poate vizualiza zonele de alocări locale și globale. O altă facilitate este oferită de **breakpoint**-uri bazate pe mesaje Windows recepționate de către aplicația dumneavoastră.

Notă: În continuare, autorul articolului face o analiză amănunțită a facilităților oferite de produs și a performanțelor sale. Pentru marea majoritate a criteriilor de analiză s-a acordat „Excelent", doar cîteva puncte fiind cotate cu „Foarte bun". În aceste condiții scorul final este 9.0, valoare foarte greu acordată în general.

De aceea acest produs a primit ștampila „Buyers Assurance Seal — Recommended Product" care reprezintă atingerea unui standard la care satisfacția cumpărătorului este asigurată. Vrem, totuși, să facem o remarcă pentru potențialii utiliatori din România. Cerințele hardware sînt destul de ridicate:

— echipamente bazate pe 286, 386, 486;

— 1 MB de memorie extinsă pentru aplicații Windows;

— 15 MB spațiu de memorie pe disc.

Traducere și adaptare
Eugen GEORGESCU

MS—DOS (II)

(Urmare din pag. 31)

● Exemple:
Pentru copierea unui fişier numit bloc.typ, din directorul de lucru în directorul rădăcină, de pe discul C:, sub numele clădire.sav, se foloseşte comanda: copy bloc.typ c:\clădire.sav.
Comanda
copy dosar 1.txt + dosar 2.txt + b: dosar.txt raport.doc combină fişierele numite dosar 1.txt, dosar 2.txt şi b:dosar.txt şi le pune în fişierul raport.doc în directorul de lucru. Dacă s-ar fi omis fişierul destinaţie, MS-DOS ar fi memorat rezultatul concatenării sub numele primului fişier. Fişierul destinaţie este creat cu data şi timpul curent.
Un exemplu de copiere a tuturor fişierelor din directorul c:\nivel în rădăcina discului A:, care foloseşte o comandă prescurtată, este:
copy c:\nivel A:

13. **Ctty** I
● Scop: Permite schimbarea perifericului de unde se trimit comenzi.
● Sintaxă:
ctty periferic
unde periferic specifică unitatea de intrare/ieşire de unde se vor da comenzi MS-DOS.
● Comentarii: Ctty este utilă dacă se doreşte schimbarea perifericului la care se lucrează.
● Exemplu:
Următoarea comandă mută toate comenzile de intrare/ieşire de la perifericul curent (consola) la un port AUX, ca de pildă un alt terminal.
ctty aux
Următoarea comandă mută înapoi intrările/ieşirile la ecranul şi tastatura consolei.
ctty con

14. **Date** I
● Scop: Afişează sau pune data.
● Sintaxă:
date[mm-dd-yy]
● Comentarii: Numerele permise sînt
mm = 1 ÷ 12
dd = 1 ÷ 31
yy = 01 ÷ 99 sau 1980 ÷ 2079
Data, luna şi anul pot fi separate prin (—) sau (/). MS-DOS este programat să schimbe luna şi anul corect, dacă luna are 28, 29, 30 sau 31 de zile.
Această comandă iniţializează ceasul intern al calculatorului.
● Exemplu:
Comanda simplă
date
produce afişarea mesajului:
Current date is weekday mm-dd-yy
Enter new date (mm-dd-yy):—
unde weekday este ziua săptămînii scrisă în limba engleză, iar mm-dd-yy este data curentă. Dacă se doreşte schimbarea datei curente, se tipăreşte noua dată; altfel se tastează doar tasta ENTER.
Comanda
date 3—9—88
produce direct modificarea datei curente.

15. **Del (Erase)** I
● Scop: Şterge fişierele specificate
● Sintaxă:
del [disc:] numecale
sau
erase [disc:] numecale
● Comentarii: Dacă se doreşte ştergerea tuturor fişierelor dintr-un director, se foloseşte caracterul special *; prin *.* indicăm toate fişierele. În cest caz MS-DOS afişează mesajul „Are you sure?". Dacă se tastează Y (de la Yes), MS-DOS şterge toate fişierele din directorul precizat.
O dată şters un fişier, el nu mai poate fi regăsit.
● Exemplu: Comanda
del vacation.*
şterge toate fişierele cu numele vacation, aflate pe discul predefinit în directorul curent.

16. **Dir** I
● Scop: Afişează fişierele dintr-un director
● Sintaxă:
dir[disc:] [cale] [/p] [/w]
● Comentarii: Dacă nu se precizează discul sau calea , comanda dir afişează fişierele şi numele subdirectoarelor din directorul curent, de pe discul curent. Comanda dir acceptă indicatoarele:
/p Selectează modul pagină, determinînd afişarea directorului cu pauză după fiecare umplere de ecran. Pentru continuarea afişării se apasă orice tastă.
/w Selectează modul restrîns, determinînd afişarea doar a numelor fişierelor, nu şi a altor informaţii, cîte cinci fişiere pe linie.
Comanda dir, fără opţiunea /w, afişează numele fişierelor, dimensiunea lor (în octeţi), timpul şi data la care s-a efectuat ultima modificare.
Comenzile următoare sînt echivalente:
dir cu dir *.*
dir nume fişier cu dir numefişier.*
dir. ext cu dir *.ext

17. **Diskcomp** E
● Scop: Compară conţinutul discului din unitatea sursă cu conţinutul discului din unitatea destinaţie.
● Sintaxă:
diskcomp [disc 1:] [disc 2:] [/1] [/8]
unde:
disc 1 este unitatea sursă.
disc 2 este unitatea destinaţie.
● Comentarii: Diskcomp realizează compararea discurilor pistă cu pistă. Această comandă acceptă indicatoarele:
/1 Determină compararea doar a primei feţe a fiecărui disc, atunci cînd discurile folosite sînt dublă faţă.
/8 Determină compararea doar a primelor 8 sectoare ale fiecărei piste, dacă discurile conţin 9 sau 15 sectoare pe pistă.
Dacă se specifică doar un disc, se consideră drept disc destinaţie discul predefinit. Dacă se specifică acelaşi disc şi ca sursă şi ca destinaţie, diskcomp va realiza o comparare în aceeaşi unitate, cerînd introducerea succesivă a discurilor.
Dacă pistele sînt aceleaşi, diskcomp afişează mesajul:
Compare OK
Dacă pistele nu sînt la fel, diskcomp afişează un mesaj de eroare, compare, care include numărul pistei şi numărul feţei (0 sau 1) unde a fost găsită nepotrivirea. Dacă discul destinaţie nu este de acelaşi tip cu discul sursă, se afişează următorul mesaj:
Drive types or diskette types not compatible
Dacă diskcomp termină compararea, se afişează mesajul:
Compare another diskette (Y/N)?—
Dacă tastaţi Y (pentru Yes) diskcomp cere să introducem discurile dorite şi efectuează o nouă comparaţie. Dacă tastaţi N (pentru No), diskcomp se termină. Dacă discul predefinit nu conţine sistemul MS-DOS şi se termină diskcomp, veţi primi următorul mesaj:
Insert disk with COMMAND.COM în drive A and strike any key when ready.
Dacă avem o copie a unui disc făcută cu comanda copy şi dacă comparăm cu diskcomp cele două discuri, rezultatul poate fi negativ deoarece copy nu pune obligatoriu fişierele în aceeaşi poziţie pe disc.
● Exemplu:
Pentru a compara două discuri în aceeaşi unitate, folosim comanda:
diskcomp a:

18. **Diskcopy** E
● Scop: Copiază conţinutul unui disc flexibil din unitatea sursă pe un disc flexibil, formatat sau reformatat din unitatea destinaţie.
● Sintaxă:
diskcopy [disc 1:] [disc 2:] [/1]
unde disc 1 este unitatea sursă, iar disc 2 este unitatea destinaţie.
● Comentarii: Disc 1 şi disc 2 pot să fie identice. În acest caz copierea se va face într-o singură unitate de disc. Dacă se omit noţiunile de disc, MS-DOS va cere discurile. Dacă discul destinaţie nu este iniţializat, diskcopy îl iniţializează cu aceleaşi caracteristici ca şi discul sursă.
Comutatorul /1 indică copierea doar a primei feţe a discului. Dacă se omit ambele opţiuni, MS-DOS foloseşte o singură unitate, cea predefinită. Dacă se precizează doar o unitate de disc, aceasta este considerată sursă, iar cea predefinită destinaţie.
Diskcopy distruge conţinutul discului destinaţie şi funcţionează astfel:
— cere introducerea discului sursă şi aşteaptă o tastă pentru continuare;
— cere introducerea discului destinaţie şi aşteaptă o tastă pentru continuare.
— după copiere afişează mesajul:
Copy another diskette (Y/N)?—
Dacă se tastează Y se reiau operaţiile anterioare; dacă se tastează N se termină diskcopy.
Diskcopy copiază discul sursă sector cu sector.
Dacă dorim să copiem un disc şi în acelaşi timp să eliminăm fragmentarea informaţiilor de pe disc, folosim comenzile copy sau Ucopy care copiază fişierele secvenţial.
● Observaţie: Diskcopy lucrează doar cu discurile flexibile, nu şi cu discul rigid.

19. **Exe2bin** E
● Scop: Converteşte fişiere executabile de extensie
.exe în format binar.
● Sintaxă:
exe2bin [disc 1:] [numecale 1] [disc 2:] [numecale 2]
unde numecale 1 este fişierul care intră şi numecale 2 este fişierul care iese.
● Comentarii: Dacă nu se specifică extensia pentru numecale 1, se consideră .exe. Dacă nu se specifică extensia pentru numecale 2, se consideră .bin. Dacă nu se specifică disc 2, se consideră disc 2 = disc 1. Dacă nu se specifică numele celui de-al doilea fişier, se consideră ca fiind identic cu primul.

20. **Exit** I
● Scop: Iese din programul comand .com şi se întoarce într-un nivel anterior dacă există.
● Sintaxă: exit
● Comentarii: În numeroase programe de aplicaţie în timpul execuţiei se poate ieşi în procesorul de comandă MS-DOS şi apoi se poate reveni în program.

PC 386 SX

Company	Model	Price	Speed	Floppy Drive
Acer	1100SX	$2,640	16MHz	Either
	1120SX	$2,840	20MHz	Either
Acma	386SX Executive System	$1,995	16MHz	Both
	386/20MHz Pro-System	$2,395	20MHz	Both
Advanced Logic	Business VEISA 32CSX Model 40	$4,299	20MHz	5.25-in.
Research	PowerFlex 20CSX Model 40	$3,248	20MHz	3.5-in.
	SX PowerFlex	$2,194	16MHz	3.5-in.
American Mitac	MiSTATION 3S	$1,795	16MHz	3.5-in
	MPC2386E	$2960	20MHz	Both
Arche Technologies	Triumph 386SX	$2,035	16MHz	5.25-in.
	Triumph 386SX	$2,325	20MHz	5.25-in.
AT&T	6386SX/EL WGS	$3,644	16MHz	3.5-in.
Austin Computer Systems	386/SX-16 Winstation	$2,190	16MHz	Either
	386/S-20 Winstation	$2,290	20MHz	Either
Compaq Computer	Deskpro 386N Model 40	$3,498	16MHz	3.5-in.
	Deskpro 386S Model 40	$4,898	16MHz	3.5-in
	Deskpro 386s/20	$5,198	20MHz	3.5-in.
CompuAdd	316s	$1,889	16MHz	Either
	320sc	$2,189	20MHz	Either
Dell	System 316SX	$1,999	16MHz	Either
	System 320LX	$2,499	20MHz	Either
Epson	386SX Plus	$3,598	16MHz	5.25-in.
Everex	Tempo 386SX/16	$2,441	8/16MHz	5.25-in.
Gateway 2000	Gateway 386SX	$1,795	16MHz	Both
Hyundai Electronics	Super-386SE	$2,544	8/16MHz	5.25-in.
IBM Corp.	PS/2 Model 55 SX/031	$4,445	16MHz	3.5-in.
	PS/2 Model 65 SX/061	$6,245	16MHz	3.5-in.
Insight Computers	386SX/16	$1,695	16MHz	3.5-in.
	386SX/20	$1,995	20MHz	Both
Leading Technology	6800SX	$1,599	16MHz	Both
Leading Edge	D3/SX	$2,774	16MHz	Both
	D3/SX-20c	$3,174	20MHz	Both
Magnavox	MAXStation 386SX	$2,898	16MHz	Both
Micro Express	ME 386 SX	$1,499	16MHz	Either
	ME 386-SX/SL	$1,599	16MHz	5.25-in.
	ME 386-SX/20 Caching	$1,839	20MHz	Either
Mitsubishi	MP386s	$3,095	16MHz	Either
NCR Corp.	ELPC386sx	$3,494	16MHz	3.5-in.
	PC386sx	$3,694	8/16MHz	Either
	PC386sx/20	$4,394	20MHz	Either
	PC386sx/MC	$3,894	16MHz	3.5-in.
	PC386sx/MC20	$4,594	20MHz	3.5-in.
NEC	PowerMate SX Plus	$3,448	16MHz	5.25-in.
	PowerMate SX/20	$4,048	20MHz	5.25-in.
Northgate Computer	SlimLine 386SX/16	$2,299	16MHz	Both
	SlimLine 386SX/20	$2,499	20MHz	Both
Panasonic	FX-1925SCSYS	$3,148	16MHz	3.5-in.
PC Brand	386/SX-16	$1,943	16MHz	Both
	386/SX-20	$2,043	20MHz	Both
Swan Technologies	386SX-16	$1,695	16MHz	Either
	386SX-20	$1,845	20MHz	Either
Tandon	SL 386SX	$2,118	16MHz	Either
	386sx/N	$1,987	20MHz	3.5-in.
Tandy	4020 SX	$2,958	20MHz	3.5-in.
Tangent	Model 316s	$1,875	16MHz	5.25-in.
	Model 320s	$2,384	20MHz	5.25-in.
TriStar	386/SX	$1,995	16MHz	3.5-inch
USA Flex	386SX Slim Line	$1,849	16MHz	Either
Zenith Data Systems	Z-386 SX	$3,698	16MHz	3.5-in.
	Z-386 SX/20	$4,198	20MHz	3.5-in.
Zeos	386SX-16	$1,895	16MHz	5.25-in.
	386SX-20	$2,195	20MHz	5.25-in.

BUYER'S GUIDE TO

ATENȚIE!

Precizăm ca toate tabelele comparative de prețuri și caracteristici tehnice pe care le publicăm se referă la **prețurile de producător.** Acestea sînt întotdeauna mai mici — firește — decît cele de **vînzător, agent comercial** sau **revînzător.** Tabelele de fața au fost preluate din revista „BUYER'S GUIDE", numărul din mai 1991.

Hard Disk	RAM	Slots	Special Features
40MB	1MB	4	Includes Windows 3.0
40MB	1MB	4	Includes Microsoft Mouse
65MB	1MB	8	Two-year warranty
65MB	4MB	8	Two-year warranty
40MB	2MB	9	RAM expandable to 49MB EISA-bus cachesystem; 32K RAM cache
40MB	3MB	6	32K RAM cache, accommodates ALR i486 upgrade module
40MB	1MB	6	Accommodates 386SX/20 or i486 CPU upgrade module
40MB	4MB	2	One year free on-site service toll free support line
40MB	1MB	6	Includes Windows 3.0 and a mouse
44MB	1MB	5	Two-year warranty
44MB	2MB	5	Two-year warranty
40MB	1MB	3	Supports Unix, LIM 4.0, IBM 8514/A graphics
40MB	2MB	7	Includes Windows 3.0, DOS 4.01, Microsoft mouse, LIM 4.0 support, 1 yr. GE on-site service
40MB	2MB	7	See above
40MB	1MB	3	Compaq expanded memory manager network compatibility
40MB	2MB	5	Compaq expanded memory manager, network compatibility
60MB	2MB	4	4K "four-way cache, hard drives up to 650MB
40MB	1MB	5	Toll-free support 90-day Express Part or Product Exchange, includes diagnostic software
40MB	1MB	5	See above
40MB	1MB	3	Built-in VGA adapter, LIM 4.0 support, 1 yr.on-site service included
40MB	1MB	8	Supports Dell versions of OS/2 1.1 with Presentation Manager, and Unix System V
40MB	2MB	4	System software is additional
40MB	1MB	8	Includes DOS 4.01
40MB	2MB	8	Includes Windows 3.0
40MB	2MB	5	Supports EMS 4.0, includes DOS 4.01, 18-month warranty
30MB	2MB	3	
60MB	2MB	7	SCSI busmaster
40MB	4MB	7	Also available in tower, toll-free lifetime tech support
104MB	4MB	7	Same as above
40MB	2MB	6	Internal modem
40MB	2MB	6	
40MB	1MB	3	Includes HyperDOS a DOS shell and tutorial environment, Windows 3.0 PFS: Easy Start, and Lotus Works
80MB	1MB	5	Includes Lotusworks "Six-in-One integrated software
40MB	1MB	8	15-month parts and labor warranty
40MB	2MB	8	
40MB	1MB	8	
40MB	2MB	6	Includes disk caching software
40MB	1MB	2	Mechanical key lock and software for setup and diagnostics
40MB	1MB	4	
40MB	2MB	4	Supports EMO/LIM 4.0, has four programmable passwords
44MB	1MB	4	MicroChannel system, SCSI peripheral controller
40MB	2MB	4	See above
42MB	2MB	5	Includes Windows 3.0
42MB	2MB	6	Includes Windows 3.0
40MB	1MB	5	Free lifetime technical support On-site service
40MB	1MB	5	See above
40MB	1MB	3	
44MB	2MB	8	Includes mouse, Free on-site service/90 days, Free tech support 5 yr. warranty
44MB	2MB	8	See above
50MB	1MB	7	Two-year free on-site service
50MB	1MB	7	See above
40MB	1MB	4	
40MB	1MB	2	
52MB	2MB	3	Lockable case
42MB	2MB	5	Includes Windows 3.0
105MB	2MB	7	Includes SuperVGA video card
44MB	1MB	5	Includes DOS 4.01
44MB	2MB	5	Includes a mouse
40MB	2MB	5	Includes a mouse
42MB	1MB	8	Includes DOS 4.01
40MB	2MB	7	See above

THE HOME OFFICE

ATENȚIE!

Precizăm că toate tabelele comparative de prețuri și caracteristici tehnice pe care le publicăm se referă la **prețurile de producător.** Acestea sînt întotdeauna mai mici — firește — decît cele de **vînzător, agent comercial** sau **revînzător.** Tabelele de față au fost preluate din revista „BUYER'S GUIDE", numărul din mai 1991.

DINCOLO DE LIMITELE UNEI REȚELE LOCALE

Traducere și adaptare după Network World, Dec 31/Ian 7-1991, Digital News, Ian 7-1991

Deși 50% din rețelele locale sînt de tip Ethernet, iar 36% de tip Token-Ring, după 10 ani de la proiectarea lor nu există încă un mod universal de interconectare. Pare un paradox! Să analizăm mai îndeaproape acest fenomen.

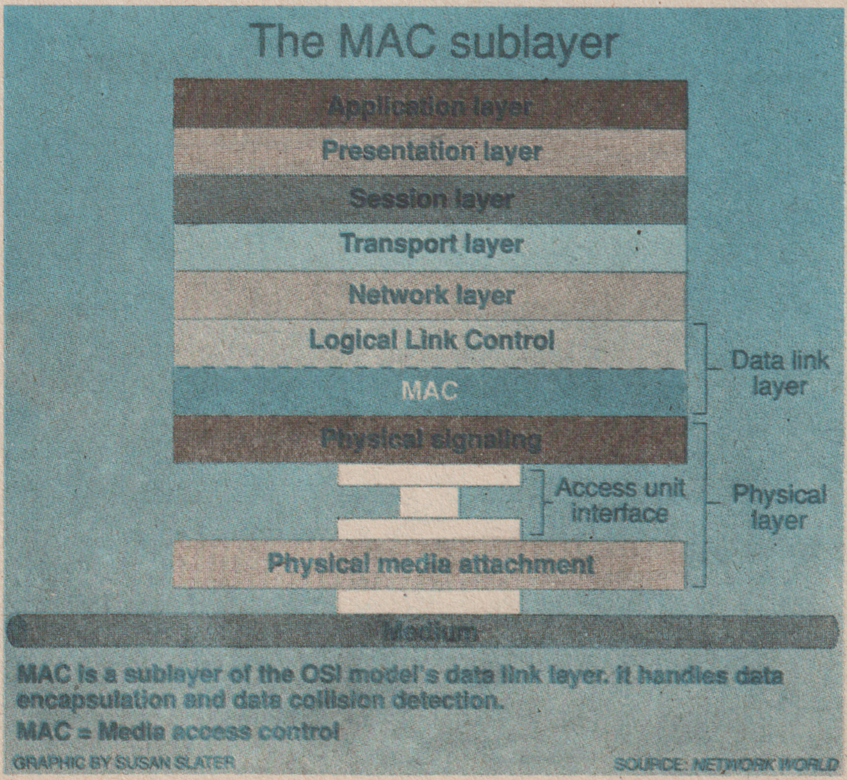
STRUCTURA GENERALĂ A UNEI REȚELE

Unii din cercetătorii care au elaborat specificația tehnică pentru rețelele de tip Novell i-au acordat o viață de 10 ani; iată că, după 10 ani, Novell a impus un adevărat standard, consfințit prin modelul ISO/OSI, ceea ce a condus la reproiectarea integrală a rețelelor cu menținerea compatibilității în ambele sensuri (cu rețele deja existente și cu cele ce se vor elabora cu noile module). Pentru o înțelegere mai ușoară a problemelor care apar, să rezumăm modelul **I**nternational **O**rganisation for **S**tandardination/**O**pen **S**ystems **I**nterconnection (**O**rganizația **I**nternațională pentru **S**tandardizare/Interconectare a **S**istemelor **D**eschise).

Modelul de referință structurează cele trei tipuri de servicii necesare într-o rețea (**P**oint-to-Point, **E**nd-to-End și **U**tilizator) în șapte niveluri (layer=strat). Primele două (Physical=fizic și Data link=legătură de date) definesc modul de comunicație necesar ca datele să fie transmise corect din punct de vedere fizic și acoperă serviciile Point-to-Point. Pentru End-to-End, s-au descris următoarele două niveluri — Network=rețea și Transport —, care definesc modul de comportare a rețelei pentru ca datele să ajungă la destinație corect din punct de vedere logic. Serviciile utilizator (User Services) acoperă ultimele trei niveluri (Session=sesiune, Presentation=prezentare și Application=aplicație). Această structură este prezentată în figura 1. Se observă că primele două niveluri sînt structurate la rîndul lor în cîte două subniveluri (Physical media attachement=adaptare fizică la mediu și Physical signaling=semnalizare fizică, respectiv Media access control = MAC = controlul accesului la mediu și Logical link control = LLC=controlul legăturii logice). Subnivelul MAC permite transferul de date conform unuia din cele trei protocoale standardizate: CSMA/CD (Carrier Sense Multiple Access with Collision Detection=acces multiplu cu sesizarea purtătoarei și detectarea conflictelor) conform standardului IEEE 802.3, TAP/BT (Token Access Protocol on a Bus Topology=protocol de acces prin secvențe de control — token — în topologie bus) conform standardului IEEE 802.4 și TAP/RT (Token Access Protocol on a Ring Topology=protocol de acces prin token în topologie inel) conform standardului IEEE 802.5.

În mai 1990, IEEE (Institute of Electrical and Electronics Engineers) prin colectivul 802.1 a aprobat un nou standard, 802.1 d, care utilizează Spanning Tree Algorithm=algoritmul de diseminare arborescentă care descrie o punte (Bridge) ce poate conecta orice tip de rețea cu orice alt tip, singura condiție fiind nerespectarea modelului 802.

Fig. 1. MODELUL DE REFERINȚĂ ISO/OSI



SOLUȚII DE INTERCONECTARE

Prima este dată de existența unor porți (Gateways) bazate pe minicomputere sau chiar calculatoare mari.

Recent, translatarea Ethernet/Token-Ring începe să fie implementată software prin opțiuni. Dar există dezavantaje majore, fie că e vorba de Gateways, fie de Routers. Ca atare s-a mai ivit o alternativă, de regulă puțin cunoscută la nivelul MAC al modelului OSI. Dacă firme cum ar fi Cross Comm Corp, IBM și Ungermann-Bass au realizat legătură la nivel de Bridge, mai nou firme precum Cabletron Systems, Digital Equipment Corp și SynOptics Communications elaborează software de cuplare la nivel MAC, utilizînd hardware-ul existent. Se obțin în acest fel „puncte centrale inteligente" în care ponderea de translatare este deținută de software și nu de hardware.

Necesitatea legăturilor Ethernet-Token-Ring este evidențiată de un raport publicat în Network World. Din 24 000 firme interogate, peste 12 000 dispuneau de calculatoare (mini, micro sau mari). Din acestea, 50% aveau rețele Ethernet, 36% Token-Ring, iar 19% ambele.

Să examinăm cîteva soluții existente:

1. **GETAWAYS** au un dezavantaj major: necesită un minicalculator sau un calculator mare dedicat pentru această activitate și sînt, de obicei, specifice unei anumite aplicații. În con-

NETWORK WORLD

Ethernet-to-token-ring MAC-layer bridges

Company	Product	Token-ring interconnections	Number of link ports	Wide-area interfaces	Filtering rate (packet/sec)	Forwarding rate (packet/sec)	Type of bridge	Standard configuration	Price
CrossComm Corp. Marlborough, Mass. (508) 481-4060	ILAN	4M bit/sec, 16M bit/sec; works with any vendor's network interface card	4	V.35, RS-232, T-1, X.21, RS-422	13,500	4,000	User option, can operate as either an encapsulation or translation bridge	2 ports standard, 2 extra ports available	$7,900
IBM Armonk, N.Y. (800) 426-2468	IBM 8209 LAN Bridge	4M bit/sec, 16M bit/sec	2	None	10,000 (for 64-byte packets)	3,000 (from Ethernet to 4M bit/sec token-ring, based on 64-byte packet size)	Translation	1 Ethernet port and 1 token-ring port	$7,445
Ungermann-Bass, Inc. Santa Clara, Calif. (408) 496-0111	Access/One Ethernet Token-Ring Data Link Bridge	4M bit/sec token-ring adapter cards from all vendors	2	None	9,000	3,900	Translation	1 Ethernet port and 1 token-ring port	$5,250

MAC — Media access control

This chart includes a representative selection of Ethernet-to-token-ring MAC-layer bridges. These vendors may offer other MAC bridges, and other vendors not included may offer a full range of competitive bridges.

SOURCE: *NETWORK WORLD*

Fig. 2. PUNŢI ETHERNET/TOKEN-RING LA NIVEL MAC

trast, un Bridge la nivel MAC nu prezintă aceste dezavantaje. Totuşi există o limitare importantă, dată de debitul posibil al reţelei (măsurat în pachete pe secundă), aşa cum se poate vedea din figura 2.

2. Există şi **ROUTERS** care realizează această conversie; sînt însă foarte scumpe (10 000—30 000 dolari) şi costul nu se justifică atunci cînd e necesară doar o simplă conversie.

3. O soluţie naturală este aceea de a include conversia protocolului în sistemul de operare al **FILE-SERVER**-ului de reţea. Dezavantajul este acela că, de regulă, protocoalele diferite implică sisteme de operare diferite, iar conversia între diverse sisteme de operare este infinit mai dificilă.

4. O altă soluţie, aceea de realizare a unui **BRIDGE**, este o soluţie pe cît de interesantă, pe atît de dificil de realizat, ca dovadă numărul mic de firme care au realizat astfel de produse. Chiar implementarea lor la nivel MAC este dificilă. Necesitînd o transparenţă totală a procesului faţă de utilizator, implică un software intensiv. Principalul obstacol este constituit de structura diferită a pachetelor (denumite frame=ferestre, adică mesaje cu structură fizică distinctă, care se transmit integral la o singură conectare între sursă şi destinaţie, dar care nu conţin în mod necesar mesaje logic întregi), după cum se poate vedea în figura 3. Deşi aminteam că modelul ISO/OSI are ca precursor reţelele Ethernet, Rich Seifert, coautor al specificaţiei Ethernet V1.0 şi V2.0 şi al specificaţiei IEEE 802.3 şi 802.4, explică această discrepanţă prin condiţiile în care a fost imaginat sistemul Ethernet: memoria era factorul tehnologic restrictiv al anilor '80, de aceea lungimea maximă a pachetului a fost limitată la 1 500 bytes. Spre deosebire de un BRIDGE fizic, cel realizat la nivel MAC îndeplineşte două sarcini suplimentare: filtrează pachetele pentru a acţiona doar asupra pachetelor destinate unor puncte externe şi realizează o conversie de frame, necesară din considerentele arătate mai sus. Se pare că problema cea mai dificilă este aceea de a diviza pachetele în procesul de comunicare dinspre TOKEN-RING spre ETHERNET. Există mai multe soluţii posibile, una foarte elegantă fiind implementată de Ungermann-Bass, denumită negocierea lungimii: dacă un pachet este prea lung, puntea retransmite pachetul spre TOKEN-RING, cerînd fragmentarea sa. Nici comunicarea ETHERNET TOKEN-RING nu este foarte simplă întrucît trebuie construită şi menţinută o tabelă de adrese care se obţine prin transmiterea unor „pachete de explorare" care ridică harta dispunerii topologice a diverselor aparate.

Conversia de protocol ridică şi ea probleme (cum am arătat şi pachetele sînt diferite, dar în mică măsură, fiind structural aproape identice). În general, conectarea prin BRIDGE la nivel MAC se poate face prin protocoale de tip TCP/IP (Transmission Control Protocol/Internet Protocol) şi IPX (Internetwork Control Protocol), dar numai unul singur la un moment dat. Mai mult, conectarea la nivel de protocol mai înalt prin MAC devine imposibilă.

În concluzie, interconectarea reţelelor nu se poate face la nivel MAC pentru translatarea de protocol TCP/IP-NET-BIOS (ca şi pentru orice altă combinaţie de protocoale), nu se poate face nici la nivel de ROUTER (care suportă un singur protocol), se poate face numai la nivel de GATEWAY, cu costuri corespunzător de mari.

5. Ultimul mod de conectare la care ne vom referi este prin **INTELLIGENT HUBS** (centru-nod-inteligent). El asigură conectivitatea la nivel fizic între Ethernet şi Token-Ring. Interesant este însă faptul că, deşi prezentate ca mod posibil de interconectare, nu o pot realiza singure, întrucît nu pot realiza nici translatare, nici cuplare (de tip Bridge). De regulă, un „centru" este reprezentat printr-o cutie de conexiuni, deci este departe de posibilităţile conexiunii la nivel MAC. Totuşi, se pare că vor apărea noduri inteligente, capabile de conexiuni Ethernet/Token-Ring, după cum anunţă firmele Cabletron şi SynOptics.

CÎTEVA CRITERII DE EVALUARE

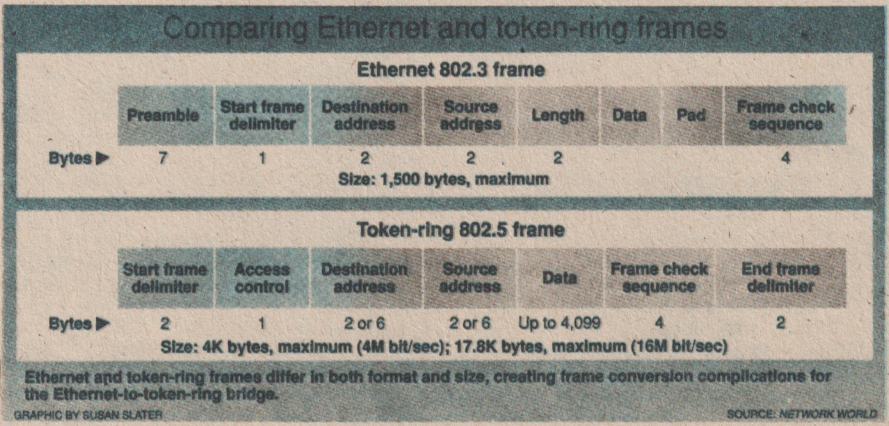
Principalul criteriu avut în vedere este viteza de transfer (debitul) măsurat în pachete/s. Debitul maxim este de 4 000 p/s (produsul ILAN al firmei Cross Comm), dar nu este mult mai mare decît cel de 3 900 p/s (Ungermann-Bass). Dacă ne referim la diferenţa de preţ de 2 650 dolari în favoarea celui de-al doilea produs, ne putem face o idee despre eficienţa unui astfel de produs. ILAN oferă mai multe interfeţe pentru conectarea reţelelor extinse şi un număr dublu de porturi de legătură (4). Referindu-ne la rata de filtrare (posibilitatea de a separa pachetele destinate reţelei locale de cele care trebuie translatate pentru o reţea eterogenă externă), cele două firme oferă produse care au rata de filtrare respectiv 12,5 k şi 9 k p/s, la o diferenţă de preţ de 5 250 dolari. Deci trebuie avute în vedere cerinţele concrete, o facilitate care nu e necesară în mod expres poate creşte preţul nejustificat.

În concluzie, popularitatea punţilor la nivel MAC va creşte pentru interconectarea reţelelor diferite care rulează acelaşi protocol. Există două tendinţe serioase: IPX se va extinde de la reţele Ethernet la Token-Ring, iar Apple Talk al calculatoarelor Macintosh va fi înlocuit cu Ethernet şi Token-Ring pentru comunicarea între calculatoare de acelaşi tip.

Utilizatorii care rulează acelaşi protocol pe reţelele Ethernet şi Token-Ring nu e necesar să cumpere ROUTERS sau GATEWAYS, ci se pot limita la BRIDGES, produse mai ieftine.

Camil SCHIAU

Fig. 3. STRUCTURA PACHETELOR ETHERNET ŞI TOKEN-RING



Computerworld

DIFERENŢA ÎNTRE MEMORIA EXPANDATĂ ŞI CEA EXTINSĂ

Traducere şi adaptare după articolul cu acelaşi titlu de Frank Fleming din revista INFOWORLD, 18 martie 1991

ÎNTREBARE:
Care este diferenţa între memoria expandată şi cea extinsă?
RĂSPUNS:
Doar foarte puţini dintre cei întrebaţi au fost capabili să răspundă limpede la această întrebare, aşa că vom da o dată pentru totdeauna o explicaţie clară asupra diferenţelor între memoria extinsă şi cea expandată în cazul calculatoarelor IBM PC şi compatibilelor.

Aşa cum probabil cunoaşteţi, un microprocesor utilizează fiecare byte de memorie folosind un număr numit adresă. Gama de valori începe de la 0, cea mai mică adresă, pînă la o adresă maximă care depinde de tipul procesorului. Primul calculator IBM PC folosea procesorul Intel 8088, un chip care era capabil să genereze doar un milion de adrese. De aceea el ar putea adresa deci maximum 1 milion de **bytes** (numit şi megabyte). Din acest spaţiu, circa 5/8, adică 640 k, a fost alocat pentru programele utilizator şi date. Aproximativ 64 k pînă la 128 k sînt utilizaţi de memoria video şi programele din **chip**-urile ROM BIOS. (Restul a fost făcut disponibil pentru BIOS-ul adiţional care este montat pe plăcile altor cuploare — FD/HDD, EGA/VGA — sau pur şi simplu a fost pierdut.)

Nu a trecut mult timp pînă ce utilizatorii au devenit total dezamăgiţi de acest aranjament. Pe măsură ce programele au început să-şi sporească necesităţile de memorie RAM, utilizatorii computerelor au început să vocifereze pentru cît mai multă memorie în care să ţină **spreadsheet**-uri cît mai mari şi fişiere pentru procesarea de texte cît mai ample. Dar, deoarece nu exista nici o cale pentru a genera mai multe adrese, singura soluţie a fost aceea a unei scheme denumită „comutare de **bank**-uri" — schemă în care diferite **chip**-uri de RAM ocupă pe rînd aceleaşi adrese.

Pentru a vizualiza cum funcţionează acest mecanism, imaginaţi spaţiul de memorie al calculatorului dumneavoastră ca o linie verticală de octeţi (bytes), avînd adresele de la 0 la 1 MB. În această schemă, comutarea de **bank**-uri este echivalentă cu alunecarea orizontală a unei „benzi" de memorie către stînga sau dreapta, permiţînd diverselor porţiuni ale ei să devină porţiuni ale benzii verticale (partea pe care o vede în mod direct microprocesorul). Dimensiunea benzii de RAM care alunecă stînga sau dreapta poate fi oricît de mare, permiţînd practic un potenţial infinit pentru sporirea dimensiunii memoriei.

Specificaţiile pentru memoria expandată (EMS — Expanded Memory Specifications) au fost publicate de firmele Lotus, Intel şi Microsoft în 1985. (De multe ori o puteţi recunoaşte sub denumirea LIM, după iniţialele celor trei companii.) Producătorii de extensii de memorie şi cei ce dezvoltau aplicaţii au început să ofere facilităţi pentru această specificaţie şi astfel cam din 1986 EMS a început să prindă. Specificaţia aceasta utilizează de preferinţă mai multe benzi de RAM (pînă la 4) în loc de una.

Memoria expandată a cîştigat piaţa în zilele de glorie ale PC-urilor bazate pe 8088. O dată cu introducerea în 1984 a modelului IBM PC AT au apărut şi posibilităţi mai bune de adresare a memoriei datorită folosirii microprocesorului 80286. În loc să aibă un spaţiu de adrese de numai 1 MB, un sistem bazat pe 80286 ar putea adresa teoretic o memorie de pînă la 16 MB fără a fi nevoie de vreun mecanism de comutare de **bank**-uri. (Nici măcar compatibilele bazate pe 386 şi 486 nu pot manipula mai mult.) IBM a denumit memoria adiţională — situată peste 1 MB — memorie **extinsă**. Memoria extinsă este deci pur şi simplu memoria care este situată peste 1 MB şi îmbunătăţeşte posibilităţile de adresare ale „bătrînului" 8088.

Au existat mai multe probleme ce au apărut înainte ca memoria extinsă să-şi cîştige popularitatea. Exceptînd o foarte îngustă bandă de 64 bytes, denumită High Memory Area (HMA), memoria extinsă poate fi utilizată doar cînd procesorul se găseşte într-un mod special, denumit „mod protejat", mod în care aplicaţiile DOS existente nu funcţionează.

Le-au trebuit foarte puţini ani programatorilor, ca şi producătorilor de echipamente, pentru a găsi soluţia comutării din/în mod protejat de o manieră elegantă. De altfel, marea majoritate a utilizatorilor încă mai au PC-uri XT. Deşi 8088 nu poate utiliza memoria extinsă, nu există nimic care să împiedice un AT sau compatibil să utilizeze memoria expandată (EMS). Chiar dacă memoria extinsă este o soluţie principial mai bună, EMS-ul a fost pentru mai mulţi ani modul cel mai popular de a sparge bariera celor 640 kB.

Această situaţie însă s-a schimbat. Astăzi maşinile bazate pe 286, 386 şi 486 şi-au luat partea leului din piaţă, utilizarea memoriei extinse a ajuns la maturitate. De fapt, anumite programe, ca, de exemplu, Windows 3.0, preferă la ora actuală să folosească memoria extinsă în loc de cea expandată.

Ca o consecinţă actuală a **manager**elor de memorie pentru 386, ca de exemplu QEMM şi 386 MAX, se poate chiar ca memoria extinsă (sau o porţiune din ea) să funcţioneze ca memorie expandată în beneficiul acelor programe care folosesc doar EMS. (Ele nu sînt deloc puţine şi chiar produse de marcă şi de ultimă oră merg pe această specificaţie!) Probabil că aţi mai auzit şi de termenul de XMS (Extended Memory Specification) folosit pentru a descrie programe care ajută la partajarea memoriei extinse între diferite programe. **Driver**-ul HIMEM.SYS de la Microsoft (MS-DOS) este un **manager** XMS, în această categorie mai existînd QEMM şi 386 MAX.

Aceasta a fost cel mult o explicaţie asupra memoriei expandate şi extinse. Din fericire sistemele de operare devin din ce în ce mai performante la capitolul management de memorie, astfel încît în curînd utilizatorul nu va mai trebui să cunoască **managementul** memoriei: software-ul va gestiona zonele disponibile şi le va utiliza pentru dumneavoastră.

Nota INFOCLUB:

În versiunea sa originală, acest articol se găseşte la rubrica „Sfatul cumpărătorului". Tot în acest sens mai există o completare.

În afară de memoria extinsă şi expandată mai există la echipamentele mai noi şi memorie ascunsă (shadow). Aceasta este o memorie RAM, mapată între 640 kB şi 1 MB, în care de regulă se transferă ROM BIOS-ul (baza plus eventualele extensii de pe cuploare) în momentul punerii sub tensiune. Sporul de viteză care se obţine este dat exact de diferenţa dintre timpul de acces al unei memorii (E)PROM şi RAM la fiecare apel al unei funcţii BIOS (citire tastatură, scriere/citire hard disk etc.).

La un calculator bazat pe microprocesorul 80286, structura hardware şi BIOS-ul (la echipamentele proiectate în ultimii 2—3 ani) oferă posibilitatea ca la operaţia numită „set-up" să luăm anumite cantităţi din memoria extinsă şi să le dăm funcţionalitate de memorie „shadow" şi/sau expandată.

La maşinile bazate pe 386 şi 486 problema memoriei expandate se rezolvă prin **driver**-e software deja standardizate, folosind modul de lucru „virtual".

Eugen GEORGESCU

Doina ISTRĂTESCU,
Eugen GEORGESCU

SQL?! Nimic mai simplu!

Răsfoind revistele de specialitate, întîlnim aceste trei inițiale din ce în ce mai des și, mai ales, legate de numele unor produse și firme de certă notorietate: Oracle (Oracle Corp.), dBase IV (Ashton Tate), SQL Link (Borland), SQL Server (Novell, Inc.). Din cele de mai sus rezultă clar că este vorba de sisteme de gestiune a bazelor de date care eventual (preferabil) lucrează într-o rețea locală. Ce este totuși SQL: un limbaj, o interfață? Răspunsul corect: un limbaj **neprocedural** folosit de regulă pentru interogarea bazelor de date.

Am subliniat cuvîntul neprocedural. Să încercăm să explicăm aceasta. Marea majoritate a limbajelor de programare sînt de tip procedural, adică pentru a rezolva o problemă, trebuie să creăm o metodologie numită algoritm pe care să o implementăm cu ajutorul unui limbaj. În cazul concret al gestiunii bazelor de date, limbajul de tip dBase ar fi exemplul clasic. Dacă dorim să servim o tranzacție de complexitate medie vom proceda cam în felul următor: analizînd rînd pe rînd înregistrările dintr-o bază, căutăm să le selectăm pe acelea ale căror cîmpuri răspund condițiilor impuse de tranzacție. Valorile obținute sînt afișate pe ecran sau păstrate în variabile de memorie în scopul formării de expresii condiționale pentru selecții în alte baze. Cei care au realizat astfel de programe știu din propria experiență că sînt mari, complexe, dezvoltarea lor este laborioasă, iar facilitățile de depanare oferite sînt sub nivelul celor oferite de alte limbaje (ex. C). Limbajele neprocedurale, în contrast cu cele enunțate mai sus, nu solicită din partea programatorului să inventeze un algoritm de rezolvare a problemei, ci doar să o expună corect, rezolvarea ei rămînînd în sarcina exclusivă a calculatorului. Astfel de limbaje au apărut de multă vreme, însă succesul comercial s-a manifestat mai de curînd și ar mai fi de remarcat că acest lucru s-a petrecut tocmai în domeniul gestiunii economice. Să încercăm să explicăm aceasta printr-un exemplu prozaic.

Să presupunem că sînteți analist-programator în cadrul unei firme. Consiliul de conducere aflat în ședință cere urgent o situație relativ complicată. Dacă folosiți o metodă procedurală, veți cere un răgaz de una sau mai multe zile pentru elaborarea și testarea programului, tranzacția cerută nefiind una din cele standard utilizate curent. Un alt coleg al dv., care utilizează însă interfața SQL, va cere 5 minute pentru gîndire, 1 minut pentru tastarea cererii și 10 minute pentru prelucrare și imprimare. Consecințele pe care le veți suporta nu fac obiectul articolului de față!

Această flexibilitate și operativitate au făcut ca interfața SQL să fie introdusă la unele produse de largă răspîndire (dBase IV) sau să fie unica metodă de lucru la altele de aceeași notorietate. S-a ajuns chiar la construcția unor supercalculatoare la care limbajul SQL să fie inclus. Totuși nevoia de procedural datorită caracterului secvențial controlabil se manifestă la construcția interfețelor de intrare și din acest motiv există unele produse care rezolvă acest deziderat prin grefarea pe un limbaj procedural (de regulă C) cu ajutorul unor biblioteci. Un bun exemplu ar fi componenta PRO—C de la sistemul de gestiune Oracle.

Limbajul SQL este destinat bazelor de date cu organizare de tip relațional. Totuși există și o clauză SQL care permite o imagine de tip ierarhic. Datele sînt organizate în tabele la care coloanele sînt cîmpurile iar liniile sînt înregistrările. De regulă, sistemele de gestiune a bazelor de date (SGBD) acceptă cel puțin cîmpuri de tip numeric, text și dată calendaristică. Pentru a se putea opera cu ele, acestea vor primi denumiri simbolice sugestive. Tot astfel de denumiri vor primi și tabelele.

Pentru a putea înțelege mai ușor esența limbajului vom porni de la un exemplu uzual. Să presupunem că în cadrul unei instituții situația personalului este înscrisă într-un tabel ce poartă denumirea angajați și cuprinde următoarele cîmpuri: marcă, nume, prenume, funcție, secție, salariu, data (angajării).

Să presupunem că tranzacția cerută este: lista cu numele, prenumele și salariul tuturor angajaților din anul 1991. În limbaj SQL acest lucru se scrie astfel:

```
SELECT nume, prenume, salariu
FROM angajați
WHERE data > 1 Ian. 1991´;
```

Listele din clauzele SELECT și FROM indică cîmpurile ce urmează a fi afișate și respectiv tabelele din care se iau acestea. Clauza WHERE conține o expresie relațională ce selectează din mulțimea înregistrărilor pe acelea care răspund condițiilor impuse (vezi figura 1).

Esența acestei cereri trebuie să o percepeți ca o intersecție de trei mulțimi: tabele, cîmpuri, înregistrări. Cu aspectul datelor la ieșire nu trebuie să vă faceți nici o problemă fiindcă majoritatea implementărilor au formate implicite mulțumitor de estetice și o mulțime de alte facilități de formatare a rapoartelor.

Se poate remarca că o astfel de cerere oferă aproape sigur și o serie de soluții cu valori identice pe cîmpuri. Rezolvarea unor astfel de situații se poate face aplicînd un modificator asupra unei cereri de cîmp.

Se presupune o tranzacție de tipul: care sînt funcțiile care sînt retribuite cu valori între 3 000 și 5 000 lei. Cererea SQL asociată:

```
SELECT DISTINCT funcție
FROM angajați
WHERE salariu BETWEEN 3 000 AND 5 000;
```

În acest exemplu ne putem concentra și asupra expresiei din clauza WHERE. Ea este o expresie relațională de comparații. Comparațiile se pot realiza atît între cîmpuri și constante, cît și între cîmpuri care pot fi situate și în tabele diferite. Pentru comparație se utilizează un set de operatori standard: >, <=, <, >=, = și != . Aceste comparații pot fi grupate după reguli de asociativitate cu ajutorul operatorilor booleeni: AND, OR și NOT.

Cu acest set de operatori se pot realiza comparații punctuale, adică între elemente individuale. În cazul SGBD-urilor acest lucru este insuficient și de aceea mai există operatori pentru comparații cu mulțimi, atît pentru mulțimi continue (BETWEEN), cît și pentru mulțimi discrete (IN). Pentru acest ultim caz vom presupune tranzacția: care sînt persoanele care sînt ingineri, cercetători sau analiști și care este salariul lor. Cererea SQL asociată:

```
SELECT nume, prenume, funcție
FROM angajați
WHERE funcție IN ('inginer', 'cercetător', 'analist');
```

După cum puteți remarca, comparațiile cu cîmpuri de tip text sînt foarte frecvente. Pentru a le spori puterea se pot utiliza două metacaractere care au următoarea semnificație:

% = orice caracter sau șir de caractere,

- = orice caracter dar numai unul.

Pentru comparația cu aceste texte generice nu se mai folosește operatorul = ci operatorul LIKE.

Pînă la acest moment concluzia pe care o tragem este că la clauza SELECT avem listă de cîmpuri, la clauza FROM avem listă de tabele și la clauza WHERE avem expresie relațională. Facilitățile limbajului sînt însă ceva mai bogate. La clauza SELECT putem avea însă o listă de expresii realizate pe baza unor operatori aritmetici, pentru cîmpuri numerice, operatori de concatenare pentru cîmpuri de tip text și aritmetici, dar specifici pentru cîmpuri de tip dată calendaristică. Expresiile pot fi, de asemenea, realizate și cu o lungă serie de funcții (ex. modulo, ridicare la putere etc.). Aceste funcții pot fi caracteristice unui tip de date sau pot fi funcții de conversie între tipuri. Asupra acestui capitol nu are rost să insistăm fiindcă fiecare implementare SQL are propriile facilități și în acest caz avantajul pare a fi de partea dBASE IV, care acceptă toate funcțiile din modul de operare procedural.

Pentru a formata rapoartele pe verticală putem să realizăm ordonări și grupări. Pentru aceste două puternice opțiuni avem la dispoziție două clauze: ORDER BY și GROUP BY.

(Continuare în pag. 42)

MAC vs PC

(Versiune prescurtată) *Jim HEID (Macworld, martie 1991)*

Unii oameni afirmă că a compara Mac-urile și PC-urile este ca și cum ai compara merele cu portocalele. Adevărul este că această comparaţie este mai degrabă între mere și aproape toate celelalte fructe. Lumea PC-urilor reprezintă această imensitate: aproximativ 71,8 milioane de mașini comparate cu aproximativ 4,4 milioane de Mac-uri. De asemenea, pentru PC-uri sînt zeci și zeci de producători, nu numai unul, sute de modele de calculatoare de cele mai diverse performanțe și prețuri de la cîteva sute la peste 15 000 $. Există mai multe sisteme de operare, patru formate de disc, patru arhitecturi pentru slot-urile de extensie și circa jumătate de duzină de standarde video. Este un adevărat corn al abundenței, dar dominat de anarhie care poate să însemne orice, de la portabile de cîteva kilograme pînă la supermașini superscumpe. În aceste condiții o comparație directă între cîteva modele de mașini este atît dificilă, cît și nesemnificativă. Din acest motiv nu ne vom opri decît asupra unor caracteristici principale.

ACCESIBILITATEA SISTEME

Pînă în octombrie trecut, prețul unui Mac nu se putea compara cu cel al unui PC. Azi, datorită lansării pe piață a trei noi modele cu preț scăzut, prețurile practicate de Apple au devenit chiar favorabile acelora practicate de doi lideri ai PC-urilor: IBM și COMPAQ.

Totuși, toate Mac-urile au în configurație standard mult mai multe facilități decît PC-urile:

— cuplor de rețea;

— cuplor SCSI (Small Computer Systems Interface);

— sistem de procesare audio;

— coprocesor matematic (excepție Mac II si).

Toate acestea sînt opționale la PC-uri, iar prețurile practicate nu sînt tocmai mici: firma Compaq adaugă 1 499 $ pentru coprocesorul 80387/33 MHz. În aceste condiții prețurile Mac-urilor încep să devină rezonabile.

Pe de altă parte există în lumea PC-urilor o piață largă de mașini ieftine, mai ieftine decît cei 999 $ ai unui Mac Classic. Dar aceste mașini sînt mai degrabă mediocre, iar cînd se încearcă rularea Microsoft Windows, care dă PC-urilor o interfață asemănătoare cu a Mac-ului, performanțele se prăbușesc cu totul.

Pe nivelul inferior se poate remarca că piața PC-urilor este mult mai bogată, dar în configurație mai săracă față de Mac. Comparînd totuși două modele cu echipări similare (IBM PS/I și Mac Classic), avantajul de preț se înclină de partea PC-urilor, deși unele componente nu suferă comparație: display-ul neclar al celui mai bun model din seria PS/I cu cel standard de la Macintosh.

În zona de mijloc, între 2 500 și 5 000 $ avantajul pare a înclina de partea Mac-ului chiar dacă arhitectura este mai puțin flexibilă. Modelele medii de PC-uri (ex.: IBM PS/2 Model 55Sx, Compaq 386/20, AST Bravo 386 Sx) folosesc un bus extern de date pe 16 biți care limitează performanțele ca și la Mac LC, dar Mac Se/30 și II si sînt computere adevărate pe 32 de biți.

Pe măsură ce mergem spre vîrf, raportul performanțe/preț crește spectaculos în favoarea Mac-urilor. S-au comparat modelul Mac II fx (4 MB RAM, hard disc 80 MB, Macintosh Display card 4 · 8 și monitor de înaltă rezoluție), oferit la 11 516 $, cu un similar IBM bazat pe un procesor mai lent 80386/25 MHz, oferit la 11 747 $. Este mai dificil să se compare o mașină cu 486/25 MHz, dar cum unele teste au arătat că performanțele cresc doar cu 15% față de 386/33 MHz, încă se mai poate afirma că Mac II fx oferă performanțe comparabile — dacă nu chiar mai bune — la un preț competitiv.

La capitolul portabile, lumea PC-urilor este net superioară, oferind și mașini puternice, dar și mașini ușoare: 386 Sx/20 MHz cu fax/modem-uri la mai puțin de 2,5 kg.

Ar mai fi de remarcat că a compara prețurile Mac-urilor numai cu ale IBM-urilor este ca și cum am compara Mercedesul doar cu BMW-ul. De fapt în lumea PC-urilor există așa-numitele **„clones"** la prețuri incomparabil mai scăzute.

PERIFERICE

Datorită competiției și pieței mult mai largi, lumea PC-urilor oferă discuri fixe, monitoare, modem-uri și extensii de memorie net competitive sub aspectul raportului calitate/preț.

În ceea ce privește imprimantele, situația este diferită. Mac-urile folosesc de regulă imprimante cu laser ce utilizează complexul limbaj PostScript. Prin comparație, PC-urile utilizează imprimante non-PostScript, ca de exemplu Hewlett-Packard, cu **controller**-e relativ simple, dar cu aptitudini tipografice simple și fără capabilități de rețea. De aceea ele sînt mai ieftine, iar adăugarea facilităților de PostScript face prețul să explodeze.

SOFTWARE

Software-ul de Mac costă în general mai puțin. Un studiu din 1989 realizat de jurnalul Soft-Letter a determinat că prețul mediu este:

— PC: 270 $,

— MAC: 195 $.

UȘURINȚA UTILIZĂRII

Aici este punctul forte al Mac-ului: o dată ce ați învățat să utilizați un program, le-ați învățat pe toate. În cazul PC-urilor, fiecare program are interfața proprie. Această situație a fost valabilă multă vreme, dar acum PC-urile au un mod de utilizare mult mai prietenos.

CONFIGURAREA HARDWARE

Unele PC-uri ieftine încă mai necesită setarea pe plăci a unor switch-uri pentru a defini sau modifica configurația. Acest mod de lucru este actualmente complet modificat prin folosirea unor utilitare rezidente în ROM BIOS care realizează acest lucru în mod automat. Aceste facilități sînt standard în cazul Mac-ului.

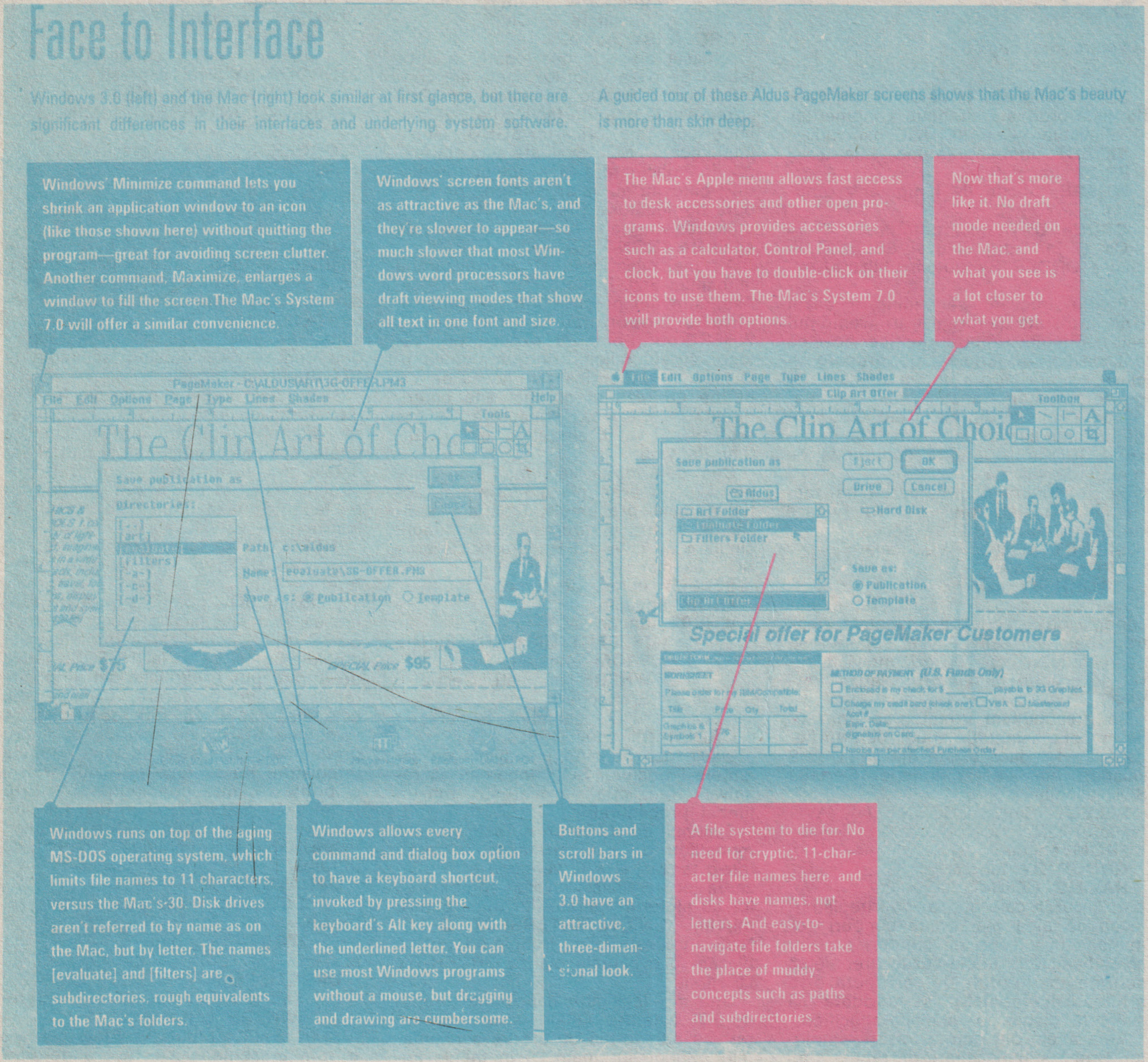
Diferența între bus-ul extern de Mac și MicroChannel de la IBM nu este prea mare. În ambele cazuri, după introducerea unei noi plăci de extensie este suficient să rulăm un simplu program de configurare prin care activăm facilitățile oferite de placă. Dar MCA este disponibil la puține mașini și acestea destul de scumpe. Competitorul său, bus-ul EISA (Extended Industry Standard Architecture) are și facilități similare de autoconfigurare.

În general la Mac configurarea constă doar în instalarea opțiunii dorite, restul realizîndu-se automat. Excepție fac Mac Plus, SE, SE/30 și Classic care necesită în cazul adăugării de memorie suplimentară și unele reglaje la sursa de alimentare, precum și alte componente interne.

OPINII ASUPRA SISTEMULUI DE OPERARE

Începătorii au nevoie de un sistem de operare simplu și ușor de învățat; veteranii doresc un sistem eficient și operare rapidă. Pornind de la acest punct de vedere putem găsi o serie de diferențe interesante.

Astfel, cea mai vizibilă componentă a sistemului de operare al Mac-ului, cea denumită „Finder", este cea mai potrivită pentru începători. Doar cîteva minute de operare cu **mouse**-ul (clicking and dragging) sînt suficiente pentru a lansa programe și a gestiona discul. Totuși facilitățile oferite de un sistem ca 7.0 sînt mult mai complexe și nece-



sită mai mult timp pentru învățare.

În lumea PC-urilor, conduce detașat sistemul de operare MS-DOS. El necesită, de regulă, pentru utilizare tastarea unor comenzi în mod linie. De asemenea, pentru operarea de zi cu zi un utilizator trebuie să cunoscă circa 7—8 comenzi împreună cu parametrii și opțiunile lor. O dată atins acest nivel, se poate lucra mult mai eficient decît pe Macintosh. Pentru începători a fost conceput un utilitar denumit DOSSHELL care apropie stilul de lucru cu cel de pe Mac.

O soluție mai bună o reprezintă produsul Microsoft Windows 3.0 care îmbogățește MS-DOS-ul cu facilități asemănătoare Mac-ului (vezi ilustrația). Totuși instalarea mediului Windows nu este chiar atît de simplă, iar pentru a obține cele mai bune performanțe trebuie să cunoaștem cele trei tipuri de memorie pe care MS-DOS-ul le accesează: standard, expandată și extinsă. Și, ca o ultimă observație, ar mai trebui să menționăm că utilizarea unor programe sub Windows le scade viteza sau chiar au incompatibilități.

Și cu toate acestea, n-am încheiat. Mai există sistemul OS/2 care permite un multitasking adevărat, spre deosebire de cel aparent realizat de MultiFinder de la Mac. Pe mașini realizate cu 386, Windows poate transforma, de asemenea, MS-DOS-ul în multitasking, dar mai pot exista probleme cu programele non-Windows.

Deși în articolul original nu se face nici o referire, nu putem să uităm pentru PC-uri sistemul de operare XEXIX care nu este altceva decît un clasic Unix System V.

APLICAȚII

La acest capitol sînt importante două criterii de comparație:
— interfața cu utilizatorul;
— intercomunicația de date.

În cazul Mac-urilor interfața cu utilizatorul este esențialmente de natură grafică și toate aplicațiile sînt unitare din acest punct de vedere. În lumea PC-urilor, Windows (MS-DOS) și Presentation Manager (OS/2) lucrează în același stil ca Mac, dar marea majoritate a aplicațiilor MS-DOS lucrează în așa-numitul regim alfanumeric, chiar dacă se folosesc menu-uri, ferestre de dialog sau suport de mouse.

Privind din punctul de vedere al celui de-al doilea criteriu, putem spune că System 7.0 de la Macintosh a realizat un mare lucru prin IAC (Inter-Application Communications) care permite schimbul de informații între aplicații „din zbor". În lumea PC-urilor această facilitate este denumită DDE (Dynamic Data Exchange) și este suportată de Windows (MS-DOS) și de Presentation Manager (OS/2).

EXTINDEREA HARDWARE

În cazul PC-urilor, slot-urile pentru cuploare sau extensiile de memorie sînt ceva comun și de obicei sînt în număr de la trei la cinci. Linia Mac compactă oferă prea puține facilități de extindere. Modelele Plus și Classic nu au nici un **slot**, iar SE și SE/30 au doar unul. Din fericire mai există și linia modulară Mac II. Modelele II ci și II cx au fiecare trei slot-uri. NuBus, II fx șase, iar II si poate avea unul, dar numai atunci cînd este

Traducere și adaptare
EUGEN GEORGESCU

(Continuare în pag. 43)

SQL

(Urmare din pag. 39)

Să presupunem următoarea tranzacție: lista ordonată alfabetic a inginerilor și analiștilor. Cererea SQL asociată:

SELECT nume, prenume, salariu
FROM angajați
WHERE funcție IN ('inginer', analist')
ORDER BY nume, prenume;

Ca și în cazul clauzei SELECT, la clauza ORDER BY putem utiliza și expresii.

Foarte frecvent se întîlnește cazul cînd la o suită de înregistrări avem unul sau mai multe cîmpuri identice. Acestea pot deveni un criteriu de grupare în clase, asupra cărora putem aplica o serie de operatori statistici: media, numărul de elemente din clasă etc. Operatorii statistici sînt de fapt funcții ce în cazul SGBD-urilor poartă denumirea de funcții de grup.

Să presupunem următoarea tranzacție: media salariului pentru fiecare funcție. Cererea SQL asociată:

SELECT funcție, AVG (salariu)
FROM angajați
GROUP BY salariu;

Așa cum clauza WHERE reprezintă condiție de selecție pe orizontală, pentru tabele, tot așa la nivel superior se poate face selecție pe orizontală la nivel de grupuri cu ajutorul clauzei HAVING.

Să presupunem următoarea tranzacție: media salariului anual pentru fiecare funcție dacă există mai mult de 2 salariați cu aceeași funcție. Cererea SQL asociată:

SELECT funcție, COUNT (*), AVG (salariu)*12
FROM angajați
GROUP BY funcție
HAVING COUNT (*) > 2;

În cazul în care o serie de înregistrări încep să aibă mai multe cîmpuri comune sau o serie de cîmpuri nu sînt semnificative în unele cazuri, trebuie să schimbăm modul de proiectare a arhitecturii tabelelor și totodată să avem la dispoziție facilități specifice.

Vom apela din nou la forța exemplului. Prima situație apare de regulă cînd valorile unor cîmpuri sînt direct determinate de valorile unui alt cîmp. Să presupunem că fiecare din secțiile ce au o codificare numerică în tabelul angajați au o serie de caracteristici: profil, localitate și spor procentual de salariu. Dacă introducem aceste trei noi cîmpuri, vom avea o lungă serie de date care se tot repetă fiindcă numărul de secții este drastic mai redus decît numărul de angajați. În aceste condiții este preferabil să creăm o tabelă numită **S** care să conțină cîmpurile: secția, profil, localitate, spor. Cele două tabele au în comun cîmpul secției care permite punerea lor în corespondență. Cum putem însă individualiza acest cîmp într-o tabelă sau alta? Foarte simplu, prefixăm acele nume de cîmpuri care se găsesc în mai multe tabele cu numele tabelei, de o manieră perfect analogă cu adresarea unui cîmp dintr-o structură în limbaj C.

Să presupunem următoarea tranzacție: numele, prenumele și salariul total al inginerilor din Cluj. Cererea SQL asociată.

SELECT nume, prenume, salariu + spor
FROM angajați, S
WHERE angajați. Secție = S. secție AND
funcție = 'inginer' AND
localitate = 'Cluj'
ORDER BY nume;

Această operație de „legare" a unor tabele pe baza valorilor egale ale cîmpului comun poartă denumirea de JOIN și reprezintă una din facilitățile cele mai importante ale oricărui SGBD. Unele implementări SQL evaluate acceptă luarea în considerare și a acelor înregistrări care nu au valori comune pe cîmpuri de legătură (OUTER—JOIN) sau utilizarea unei alte condiții decît cea de egalitate (NON—EQUI—JOIN).

Alt mecanism evoluat de utilizare a limbajului SQL îl reprezintă subcererea. Așa cum am precizat într-un paragraf anterior, operanzii dintr-o clauză WHERE pot fi precizați explicit, prin valoare, cît și implicit, prin nume de cîmp. O facilitate puternică este aceea de a îi obține ca rezultat al unei alte cereri, numită în acest caz subcerere. Se pot returna atît valori singulare, cît și mulțimi de valori.

Cea de-a doua versiune, mai complexă, va fi exemplificată prin tranzacția: numele, prenumele, funcția și sporul de salariu pentru cei ce au funcții similare cu cei din București. Cererea SQL asociată:

SELECT nume, prenume, spor
FROM angajați
WHERE funcție IN
(**SELECT** DISTINCT funcție
FROM angajați, S
WHERE angajați. Secție = S. Secție
AND localitate = 'București')
ORDER BY spor DESC;

Dacă încă nu sînteți convinși de valențele limbajului SQL, încercați să rezolvați tranzacția de mai sus într-un limbaj procedural!

Sistemul de subcereri nu este singura modalitate de a folosi mai multe cereri într-o tanzacție. Altă versiune este utilizarea lor pe același nivel și combinarea lor cu ajutorul operatorilor tipici pentru mulțimi: UNION, INTERSECT și MINUS. În acest caz nu trebuie uitat ca să plasăm o eventuală clauză ORDER BY la sfîrșit.

Una din facilitățile fascinante oferite de SQL este posibilitatea structurării ierarhice sub formă de arbore a rezultatelor. Să presupunem că în tabela angajați se mai introduce cîmpul ms care ar reprezenta „marca șefului direct". Pentru a construi arborele, pornind dintr-un anumit nod, în direcția dorită, limbajul SQL oferă două clauze:

CONNECT BY cîmp = cîmp
START WITH condiție

În prima clauză se compară cîmpurile ce au semnificația de marcă proprie și a superiorului direct. Există un operator, numit PRIOR, care dacă se aplică mărcii curente parcurgem arborele de la nod spre extremități, iar dacă se aplică mărcii superiorului parcurgerea are loc de la nod spre rădăcină.

Cea de-a doua clauză selectează înregistrarea care este considerată.

Să presupunem următoarea tranzacție: cine sînt toți angajații care sînt subordonați lui Eugen. Cererea SQL asociată:

SELECT nume, prenume, funcție
FROM angajați
WHERE prenume = 'Eugen'
CONNECT BY PRIOR marcă = ms
START WITH prenume = 'Eugen'
ORDER BY salariu;

După acest ultracondensat expozeu putem să dăm o sintaxă caracteristică marii majorități a implementărilor SQL:

SELECT [DISTINCT] listă {cîmpuri| expresii}
FROM listă tabele
[WHERE condiție]
[CONNECT BY condiție
[START WITH condiție]]
[GROUP BY listă expresii
[HAVING condiție]]
[{UNION | INTERSECT | MINUS}
SELECT --------]
[ORDER BY listă expresii]

unde:
[---] = opțional;
{ } = obligatoriu;
| = sau.

Din păcate, limbajul SQL utilizat la interfețe de intrare nu este prea comod. De aceea de obicei el lucrează fie în tandem cu generatoare speciale de interfețe de intrare, fie se grefează pe un limbaj de nivel înalt, de largă răspîndire, cum este C, folosind facilitățile de I/O oferite de acesta.

Să studiem totuși o parte din facilitățile oferite de limbajul SQL. Introducerea de noi înregistrări într-un tabel se poate face atît complet, cît și parțial, iar datele pot fi furnizate explicit în instrucțiune, cît și pot fi extrase dintr-un alt tabel. Sintaxa unui astfel de tip de instrucțiune este:

INSERT INTO tabel [(cîmp, cîmp,--)]
{**VALUES** (valoare, valoare,--) cerere}

Limbajul SQL poate asigura de asemenea și o altă facilitate caracteristică SGBD-urilor: actualizarea (modificarea) datelor. La fel ca și la INSERT, datele de intrare pot fi explicite sau implicite rezultînd dintr-o cerere. Caracteristic este însă faptul că spre deosebire de versiunile SGBD cu metode de lucru procedurale, aici selecția unei anumite înregistrări nu se face prin numărul ei ci printr-o clauză condițională WHERE la fel ca la SELECT. Dăm mai jos sintaxa pentru cele două moduri de specificare, explicită și implicită, a datelor de intrare.

UPDATE tabel
SET cîmp = expresie, cîmp = expresie,--
[**WHERE** condiție]

și

UPDATE tabel
SET (cîmp, cîmp,--) = (subcerere)
[**WHERE** condiție]

Și în sfîrșit o ultimă facilitate pe care o vom prezenta aici este cea de ștergere a înregistrărilor. Identificarea lor se va face ca și pînă acum, printr-o condiție într-o clauză WHERE. Sintaxa SQL:

DELETE FROM tabel
[**WHERE** condiție];

Toate cele prezentate anterior reprezintă doar partea de exploatare curentă din limbajul SQL. Sintaxa este dată într-o formă minimală deoarece funcție de implementare mai există o serie de extensii reprezentate printr-un set de instrucțuni complexe pentru specificarea formatului de ieșire și, de asemenea, printr-o serie de funcții ce acționează asupra valorilor cîmpurilor.

În afara părții destinate manipulării datelor, limbajul SQL posedă facilități privind administrarea tabelelor (CREATE, DESCRIBE, ALTER, DROP), facilități pentru manipularea unor informații conexe, dar de mare importanță (SPACE, INDEX, CLUSTER, VIEW, SYNONIM), precum și facilități privind securitatea accesului la date (GRANT).

În încheiere ne exprimăm speranța că sînteți convinși că operarea datelor sub formă de mulțimi (esența limbajului SQL) este incomparabil mai eficientă și mai ușoară decît manipularea secvențială a cîmpurilor și înregistrărilor.

MAC vs PC

(Urmare din pag. 41)

echipat cu Apple's NuBus Adapter Card. De asemenea, modelele II si și II fx și acces direct la semnalele procesorului prin PDS (Processor-Direct Slot).

Din acest motiv lumea PC-urilor abundă de tot felul de plăci de extensie, în special pentru standardul ISA (Industry Standard Architecture). Orice enumerare ar fi mai mult decît incompletă: cuploare video, seriale/modem/fax, adaptoare audio digitale, adaptoare pentru dispozitive speciale (scanner) etc.

În cazul Mac-urilor oferta este mai redusă și sînt mult mai scumpe. De regulă este vorba de plăci video de mare performanță care pot controla mai multe monitoare simultan (ceea ce PC-urile nu prea pot face). Mai există și modemuri, dar trebuie să remarcăm plăcile Sound Accelerator și Audiomedia de la Digidesign care transformă Mac II-urile într-o stație de lucru audio digitală fără pereche în lumea PC-urilor.

Flexibilitatea unei configurații prin slot-uri de extensie nu este unul din „punctele tari" ale Mac-urilor. Totuși ele au specific așa-numita operație de „upgrade" cu efecte spectaculoase și care nu este posibilă la PC-uri. Astfel se poate transforma un SE într-un SE/30 cu 1 699 $ (cît un PC AT cu EGA — n.r.) sau se poate transforma un Mac II utilizat într-un II fx care este în fruntea seriei pentru numai 2 999 $. Unii oameni se vaită de prețurile acestea, dar aceasta este una din facilitățile care permit întinerirea unui produs de la capătul de jos al seriei.

LUCRUL ÎN REȚEA

În lumea afacerilor, posibilitatea mașinilor de a partaja datele și resursele scumpe (ex. imprimante laser, discuri de mare capacitate) este un puternic deziderat. În cazul rețelelor mici conduc Mac-urile, dar cînd este nevoie de rețele mari, unde viteza, securitatea și siguranța în funcționare sînt critice, atunci se apelează la PC-uri.

Toate Mac-urile au o interferență inclusă pe placă pentru o rețea Local Talk. Consecința este că e necesar doar să legăm cablurile. PC-urile nu au cuplor de rețea inclus, așa că atunci trebuie să adăugăm un cuplor de rețea care sporește prețul. Dacă trecem la rețele de mare viteză situația se inversează. Un cuplor Token-Ring sau Ethernet este mult mai ieftin pentru PC-uri. Situația este similară și în cazul software-ului de rețea. Produsele pentru Mac nu au un nivel de securitate corespunzător și nici siguranța în funcționare nu are mecanisme speciale. Nu poate să existe comparație cu PC-urile unde un produs ca Novell NetWare poate gestiona în paralel pe un **file-server** două discuri pentru a preveni orice pierdere de informație în caz de defecțiune.

PERFORMANȚE

Nu se pot compara două calculatoare fără a efectua teste cu programe comune. Rezultatele au fost sintetizate în tabelele prezentate. Dacă se compară programe care pe PC rulează sub Windows (MS-DOS) sau Presentation Manager (OS/2), avantajul este de partea Mac-ului. Totuși marea majoritate a programelor PC rulează în regim alfanumeric și atunci Mac-ul rămîne în urmă.

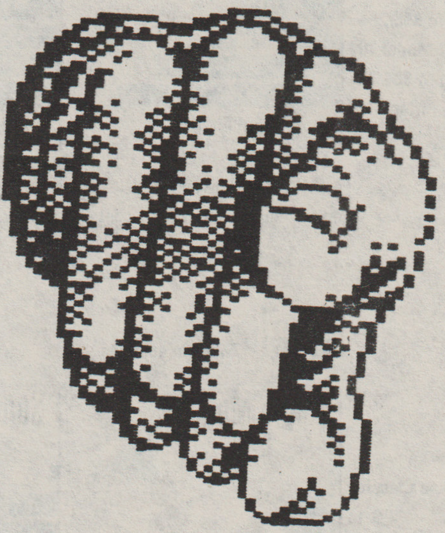
CÎȘTIGĂTORUL ESTE:

La sfîrșit putem afirma că Mac este cel mai bun calculator, dar PC-ul este versiunea optimă!

Pentru prețul unui Mac Classic cu 2 MB RAM și disc de 40 MB, mai bine luați un PC compatibil, care este mai rapid, are display color și este mult mai configurabil. El va fi mult mai avantajos pentru prelucrări de texte și gestiune economică. Dacă mai vreți să fie și portabil, este evident că-l veți găsi în lumea PC-urilor.

Dacă intrați în lumea proiectării asistate, scannării și prelucrării de imagini, audio digital și publicații color, Mac-ul este de departe cea mai bună soluție!

FIȘIERE DOS



Ordinea în care rulează fișierele DOS este mai puțin cunoscută. Pentru cei care vor să rezolve această problemă, iată și răspunsul. Sub DOS, fișierele se execută în ordinea următoare:

1) fișiere de tip .COM
2) fișiere de tip .EXE
3) fișiere de tip .BAT.

Deci un fișier de tip .COM va fi întotdeauna rulat înaintea unui fișier de tip .BAT cu același nume. De exemplu: să presupunem că într-un director există două fișiere numite TEST.COM și TEST.BAT. După cum se cunoaște, pentru fișierele cu extensiile .COM, .EXE, .BAT, lansarea în execuție se poate face invocînd doar numele, fără precizarea extensiei. În cazul de față, se va tasta doar: TEST.

Conform cu regula de mai sus, este evident că va fi rulat fișierul TEST.COM și nu cel cu extensia .BAT.

Dar ce se întîmplă dacă se specifică și o anumită extensie cînd se rulează fișierul în cauză? Răspunsul este că nu se poate rula un fișier .BAT dacă mai există un fișier .COM cu același nume în directorul curent.

Cu toate acestea, sînt și excepții, datorate în primul rînd inconsistențelor DOS, care este un sistem de operare departe de a fi perfect.

Iată de pildă, dacă DOS nu găsește un fișier cu numele specificat în directorul curent, el îl va căuta în directorii specificați în comanda PATH, dacă aceasta există. Introducînd atît numele, cît și extensia, DOS va lua în considerare întîi extensia! Deci, dacă există două fișiere TEST.BAT și TEST.COM în alți directori din PATH, tastînd TEST. BAT se va rula fișierul TEST. BAT.

Deplasîndu-ne în acel director, la o eventuală comandă TEST.BAT se va lansa în execuție, de fapt, TEST.COM.

Referindu-ne acum la extensii, iată o altă problemă interesantă. Să vedem ce s-ar întîmpla dacă ar exista un fișier numit AUTOEXEC.COM în directorul principal (root). Ei bine, DOS nu va executa decît un fișier AUTOEXEC.BAT în secvența de inițializare a sistemului. Un eventual fișier AUTOEXEC.COM este pur și simplu ignorat, fie că există sau nu fișierul cu același nume de tip .BAT. Iată deci o excepție de la regula de prioritate enunțată.

Totuși, această excepție poate fi derulată introducînd cuvîntul AUTOEXEC în fișierul AUTOEXEC.BAT. Astfel va fi căutat și rulat un fișier executabil cu numele AUTOEXEC și extensia .COM. Prin urmare, AUTOEXEC.COM se va rula mai înainte ca să se încheie. AUTOEXEC.BAT. Metoda aceasta se poate folosi, de exemplu, pentru protejarea codului din AUTOEXEC. BAT, făcîndu-l mult mai greu de modificat de către eventualii utilizatori.

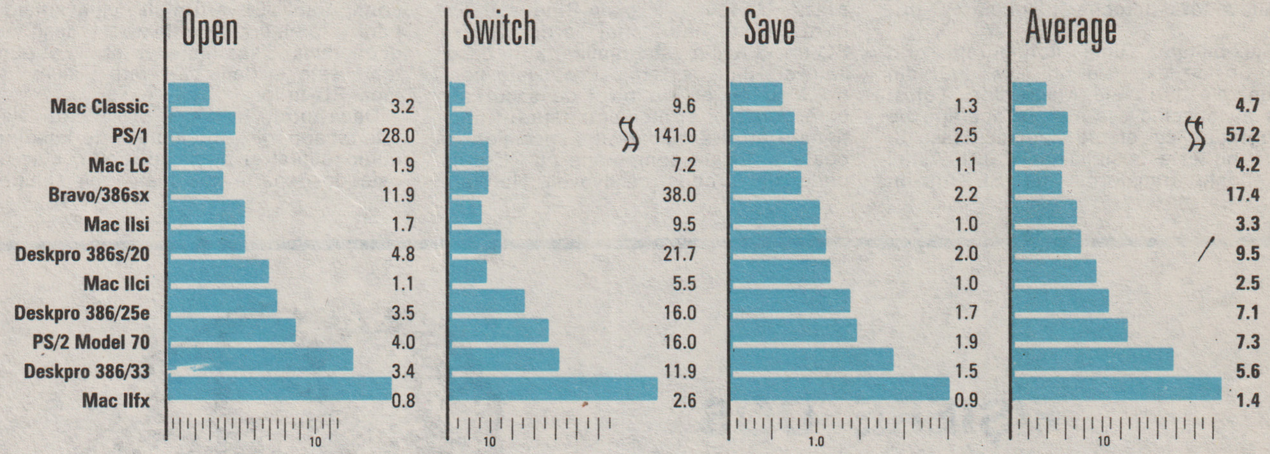
Mirel DOBRILĂ

Comparing Performance

Model	Mac Classic	PS/1	Mac LC	Bravo/386SX	Mac IIsi
Manufacturer	Apple	IBM	Apple	AST Research	Apple
RAM/hard drive configuration	2MB/40MB	1MB/30MB	2MB/40MB	2MB/45MB	5MB/80MB
Math coprocessor installed	NA	NA	NA	no	yes
System software versions[2]	6.1.6/6.0.6	4.0/3.0	6.1.7/6.0.7	3.3/3.0	6.1.6/6.0.6
Processor	68000	80286	68020	80386SX	68030
Clock rate	8MHz	10MHz	16MHz	16MHz	20MHz
Retail price as configured[3]	$1499	$1999	$2499	$2845	$4814

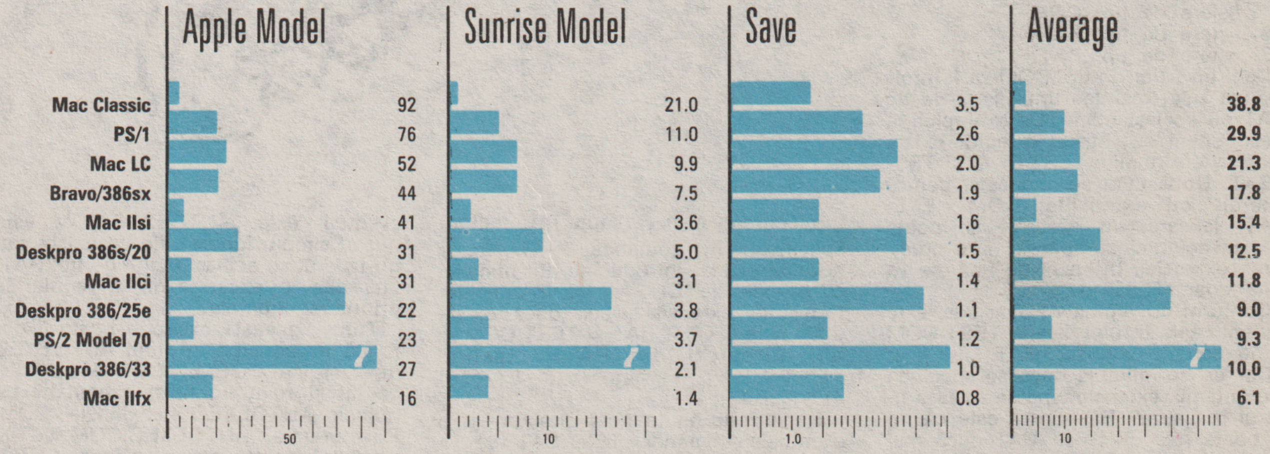
[1] Out of production, but sold while supplies last.
[2] System software versions are listed as Finder/System for Macs, and MS-DOS version/Windows version for PCs.
[3] Retail prices do not include monitors or video cards, if separate ones are required.

PowerPoint Times in seconds



For Microsoft PowerPoint I used the Columbus sample document that accompanies both PowerPoint versions. It's no contest: the Mac version left its Windows counterpart in the dust.

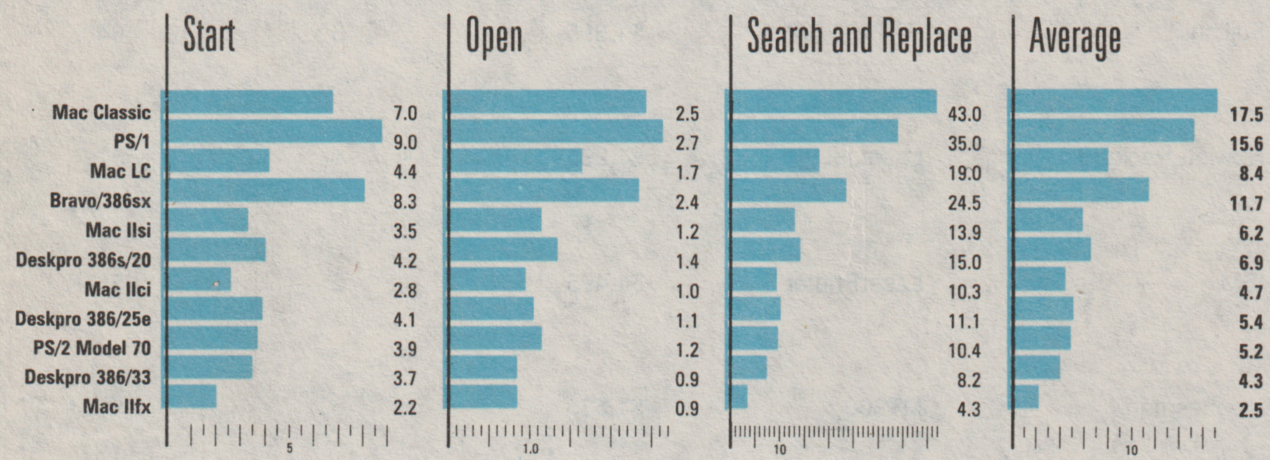
Excel Times in seconds



The Apple Model in my tests comprised a series of Excel macros, worksheets, and charts created by Apple's Ted Barnett designed to simulate an Excel work session. The Sunrise Model refers to Russell C. Anderson's calculation-intensive shareware model that graphs sunrise and sunset times and other meteorological information. The Windows version of Excel usually beat its Mac cousin—despite the fact that most of the PCs I tested lacked math coprocessor chips. Microsoft took great pains to optimize Excel for Windows, knowing it would be benchmarked against the fast, character-based Lotus 1-2-3. The effort clearly paid off.

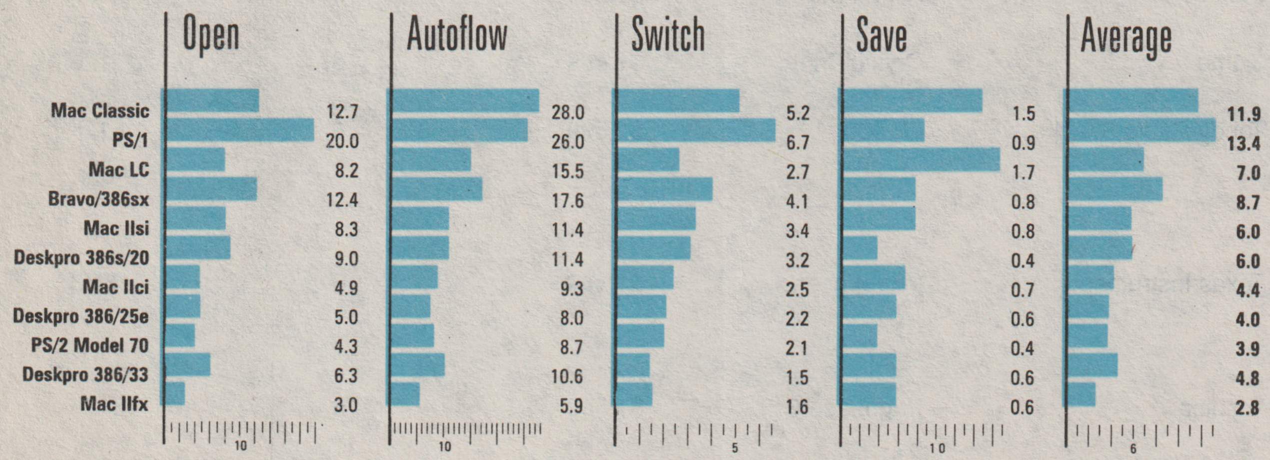
Deskpro 386s/20	Mac IIci	Deskpro 386/25e	PS/2 Model 70	Deskpro 386/33	Mac IIfx
Compaq	Apple	Compaq	IBM	Compaq	Apple
4MB/120MB	4MB/80MB	4MB/60MB	12MB/60MB	2MB/84MB	8MB/80MB
no	yes	no	no	yes	yes
4.01/3.0	6.1.5/6.0.5	4.01/3.0	3.3/3.0	4.01/3.0	6.1.4/6.0.5
80386SX	68030	80386	80386	80386	68030
20MHz	25MHz	25MHz	25MHz	33MHz	40MHz
$4299	$6669	$6199	$7220	$11,598	$10,868

Word



I used a 5-page, single-space document containing several styles of Courier. The search-and-replace test involved changing 1375 lowercase *e*'s to pound signs (#). The Mac version of Word may not have all the features of its Windows counterpart, but it's generally faster.

PageMaker



I used a 1-page PageMaker 3.0 document containing numerous bitmapped graphics and text in several styles and sizes of Helvetica. The autoflow test used a 4-page Microsoft Word document. There's no cut-and-dried winner here, although the Macs were generally faster at opening documents and switching viewing scales—two graphics-intensive tests.

Imprimante laser

COMPANY	MODEL	PRICE	SPEED (PPM)	BASE MEMORY
Brother	HL-4	$1,495	4	512K
Canon	LBP-4	$1,545	4	512K
Dataproducts	LZR 650	$1,695	6	512K
Epson	EPL-7000	$1,399	6	512K
Fujitsu	RX7100 S/2	$1,395	5	640K
Hewlett-Packard	LaserJet IIP	$1,295	4	512K
IBM	LaserPrinter E	$1,495	5	512K
Mannesman	MT906	$1,535	6	512K
Okidata	OL400	$1,000	4	512K
Okidata	OL800	$1,499	8	512K
Panasonic	KX-P4420	$1,695	8	512K
Qume	CrystalPrint Series II	$1,499	6	512K
Tandy/Radio Shack	LP 950	$1,599	6	512K
Texas Instruments	microLaser	$1,449	6	.5Mb
Toshiba	PageLaser6	$1,549	6	512K

Ghidul cumpărătorului

ATENȚIE!

Precizăm că toate tabelele comparative de prețuri și caracteristici tehnice pe care le publicăm se referă la **prețurile de producător.** Acestea sînt întotdeauna mai mici — firește — decît cele de **vînzător, agent comercial** sau **revînzător.** Tabelele de față au fost preluate din revista „BUYER'S GUIDE", numărul din mai 1991.

EMULATIONS	PAPER CAPACITY	POSTSCRIPT	SPECIAL FEATURES
HP IIP Epson FX80 IBM Proprntr Diablo 630 Brother Twinriter	1 bin 50 sheets	na	Excellent value. Generous fonts & emulations
Diablo 630 Early Canon	1 bin 50 sheets	na	Compact, reliable Software compatible modes
HP PCL Diablo 630 Epson FX80 IBM Proprntr IBM Graphics Prntr	1 bin 250 sheets	na	Multiple emulations Good paper handling
HP PCL 4 Epson FX/LQ HP IIP EPL-7500	1 bin 250 sheets	upgrade available	HP IIP compatible Low list price Upgrade to PostScript
HP PCL Diablo 630 Epson FX85 IBM Proprinter XL	1 bin 150 sheets	na	HP emulation Excellent value
HP PCL	1 bin 50 sheets	PS cartridge available	HP PCL Font capabilities Industry standard Reliable engine
HP PCL 4 IBM GL HPGL IBM PPDS	1 bin 200 sheets	2 options available	PostScript upgrade Excellent value
HP PCL MT Superset	1 bin 150 sheets	upgrade available	Jade controller PostScript upgrade
HP PCL OL400	1 bin 200 sheets	na	Excellent value HP compatibility
HP PCL OL800 Diablo 630 IBM Proprntr	1 bin 200 sheets	upgrade available	LED Page Printer Many fonts PostScript Option Excellent value
HP PCL	1 bin 250 sheets	na	Speed (8 ppm) HP compatible
HP PCL	1 bin 100 sheets	upgrade available	Liquid CrystalImaging. HP compatible
HP PCL IBM Proprntr Epson FX80 Diablo 630 IBM Graphics Prntr	1 bin 250 sheets	na	Good paper capacity HP compatible Emulations
IBM Proprntr Epson FX80 Diablo 630 IBM Graphics Prntr	250 sheets	HP compatible Emulations	
HP PCL Diablo 630 Epson FX	1 bin 250 sheets	2 upgrades available	PostScript option Apple & DOS compat.

ATENŢIE!

Precizam ca toate tabelele comparative de preţuri şi caracteristici tehnice pe care le publicam se refera la **preţurile de producător.** Acestea sînt întotdeauna mai mici — fireşte — decît cele de **vînzător, agent comercial** sau **revinzător.** Tabelele de faţa au fost preluate din revista „BUYER'S GUIDE", numarul din mai 1991.

INFOCLUB

- prima şi unica revistă de calculatoare şi informatică din România membră a IDG — International Data Group din SUA
- o unică posibilitate de a fi în legătură directă cu topul lumii informaticii
- o garanţie a succesului dv. în afacerile din lumea calculatoarelor şi informaticii — paginile de mică şi mare publicitate pe care le punem la dispoziţie

INFOCLUB INTERNAŢIONAL

- titlul noii publicaţii bilunare pe care o gîndim şi o concepem acum pentru dumneavoastră
- o publicaţie de informaţie, business şi publicitate în domeniul calculatoarelor şi informaticii
- NU UITAŢI! Septembrie este luna primelor apariţii cu care vom invada piaţa publicisticii în acest domeniu; IDG este cu noi şi de această dată!

